An Introduction to Support Vector Machine

MAP i - Computer Vision 2011/12

Jaime S. Cardoso

INESC Porto, Faculdade Engenharia, Universidade do Porto

2012/01/02

What is pattern recognition?

"The assignment of a physical object or event to one of several prespecified categories" -- Duda & Hart

- A **pattern** is an object, process or event that can be given a name.
- A **pattern class** (or category) is a set of patterns sharing common attributes and usually originating from the same source.
- During **recognition** (or **classification**) given objects are assigned to prescribed classes.
- A **classifier** is a machine which performs classification.

Examples of applications

- **Optical Character Recognition (OCR)**
- **Biometrics**
- **Diagnostic systems**
- **Military applications**

- Handwritten: sorting letters by postal code, input device for PDA‘s.
- Printed texts: reading machines for blind people, digitalization of text documents.

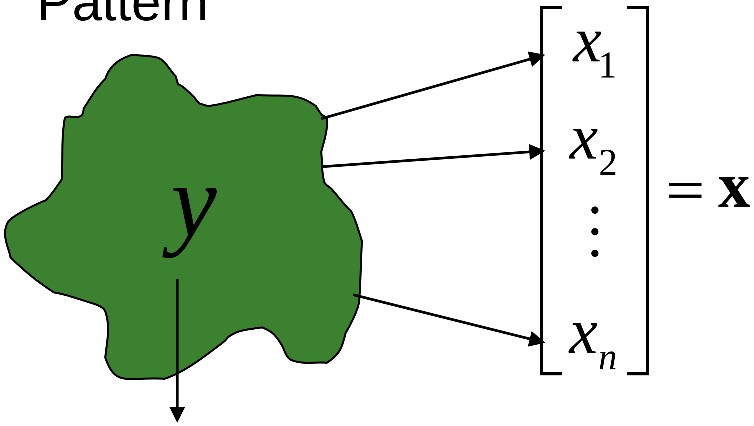
- Face recognition, verification, retrieval.
- Finger prints recognition.
- Speech recognition.

- Medical diagnosis: X-Ray, EKG analysis.
- Machine diagnostics, waster detection.

- Automated Target Recognition (ATR).
- Image segmentation and analysis (recognition from aerial or satelite photographs).

Basic concepts

Pattern



Feature vector $\mathbf{x} \in X$

- A vector of observations (measurements).

- $\mathbf{x}$ is a point in feature space X.

Hidden state $y \in Y$

- Cannot be directly measured.

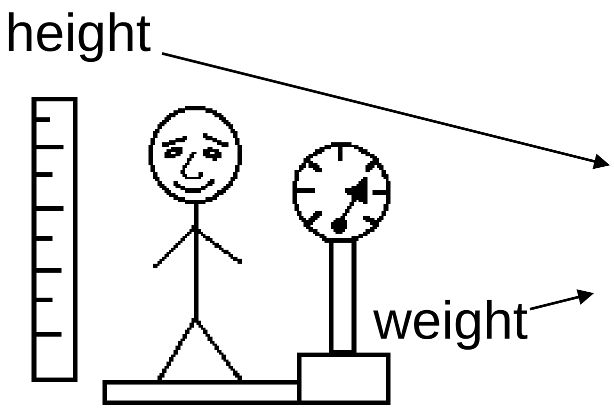
- Patterns with equal hidden state belong to the same class.

Task

- To design a classifer (decision rule) $q : X \to Y$

which decides about a hidden state based on an onbservation.

Example



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}$$

<u>Task</u>: jockey-hoopster recognition.

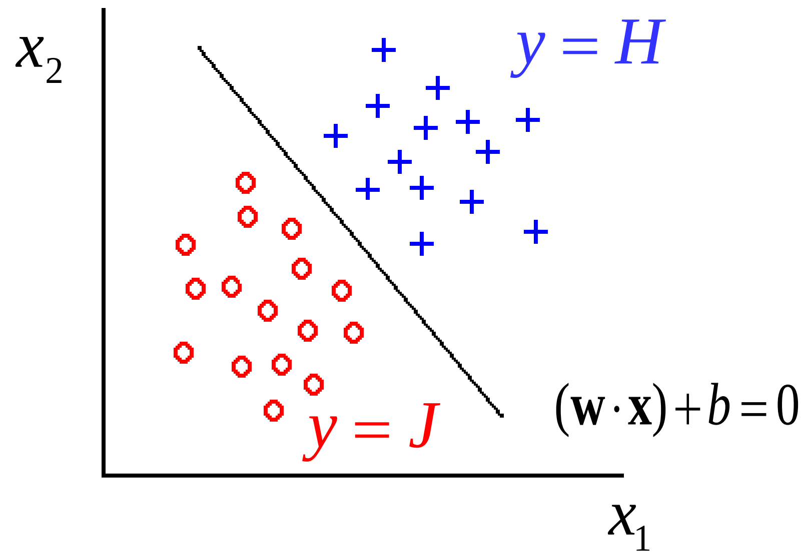
The set of hidden state is $Y = \{H, J\}$

The feature space is $X = \Re^2$

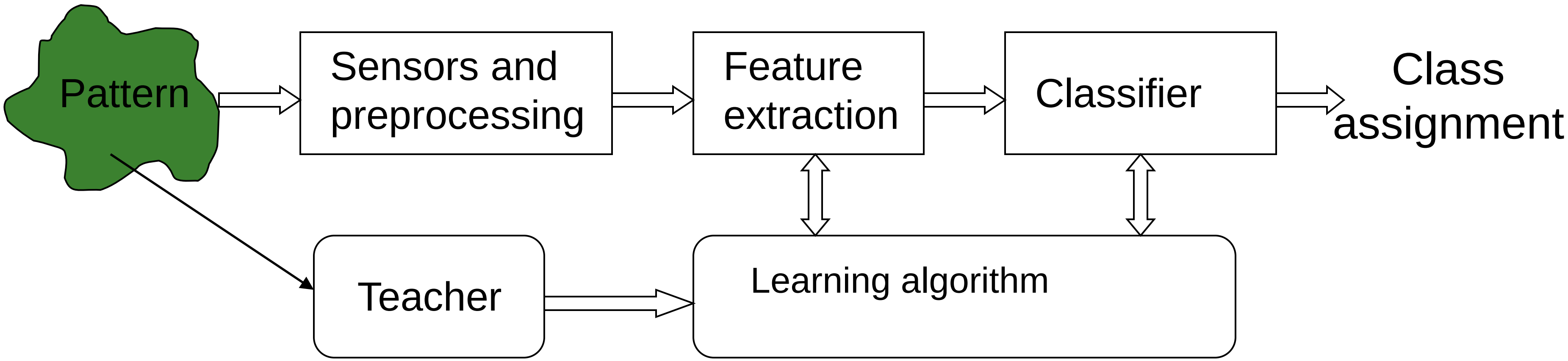
<u>Training examples</u> $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$

<u>Linear classifier:</u>

$$q(\mathbf{x}) = \begin{cases} H & if \quad (\mathbf{w} \cdot \mathbf{x}) + b \geq 0 \\ J & if \quad (\mathbf{w} \cdot \mathbf{x}) + b < 0 \end{cases}$$



Components of PR system

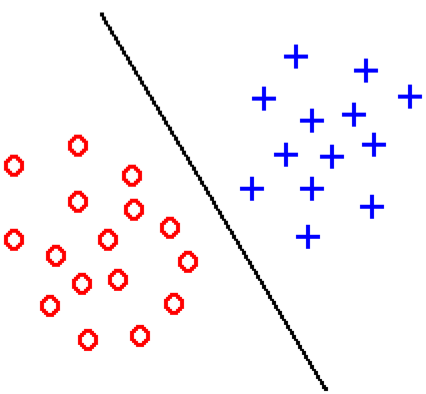


- **Sensors and preprocessing**.
- **A feature extraction** aims to create discriminative features good for classification.
- **A classifier**.
- **A teacher** provides information about hidden state -- supervised learning.
- **A learning algorithm** sets PR from training examples.

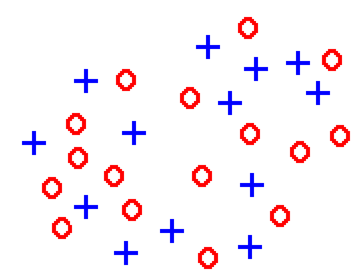
Feature extraction

Task: to extract features which are good for classification.

Good features:
- Objects from the same class have similar feature values.
- Objects from different classes have different values.



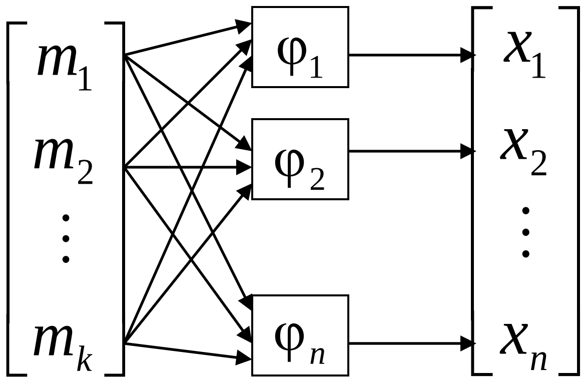
“Good” features



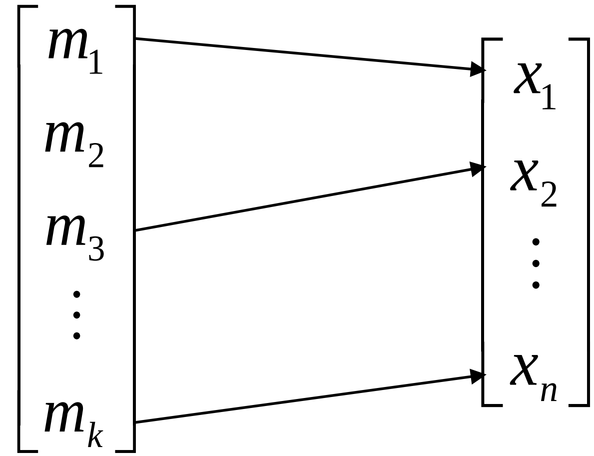
“Bad” features

Feature extraction methods

Feature extraction



Feature selection



Problem can be expressed as optimization of parameters of feature extractor $\varphi(\theta)$

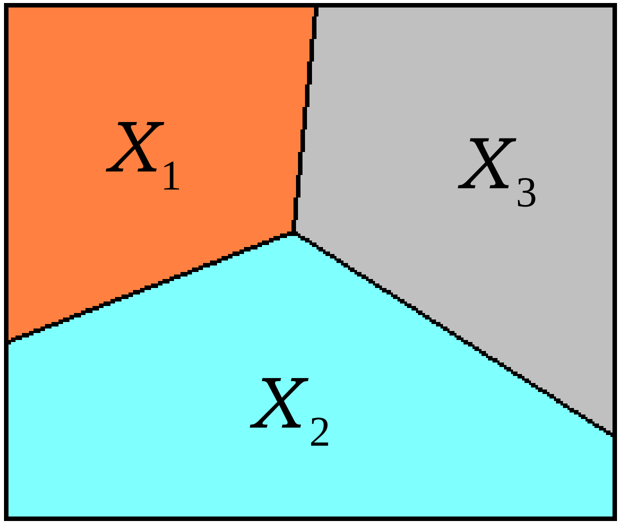
Supervised methods: objective function is a criterion of separability (discriminability) of labeled examples, e.g., linear discriminat analysis (LDA).

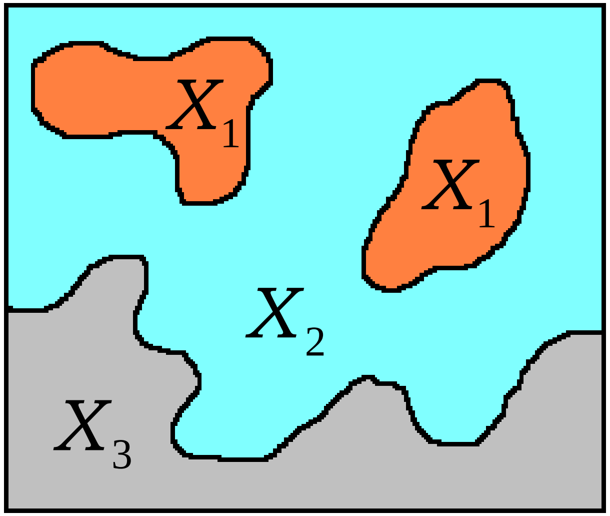
Unsupervised methods: lower dimensional representation which preserves important characteristics of input data is sought for, e.g., principal component analysis (PCA).

Classifier

A classifier partitions feature space *X* into **class-labeled regions** such that

$$X = X_1 \cup X_2 \cup \ldots \cup X_{|Y|} \quad \text{and} \quad X_1 \cap X_2 \cap \ldots \cap X_{|Y|} = \{0\}$$





The classification consists of determining to which region a feature vector **x** belongs to.

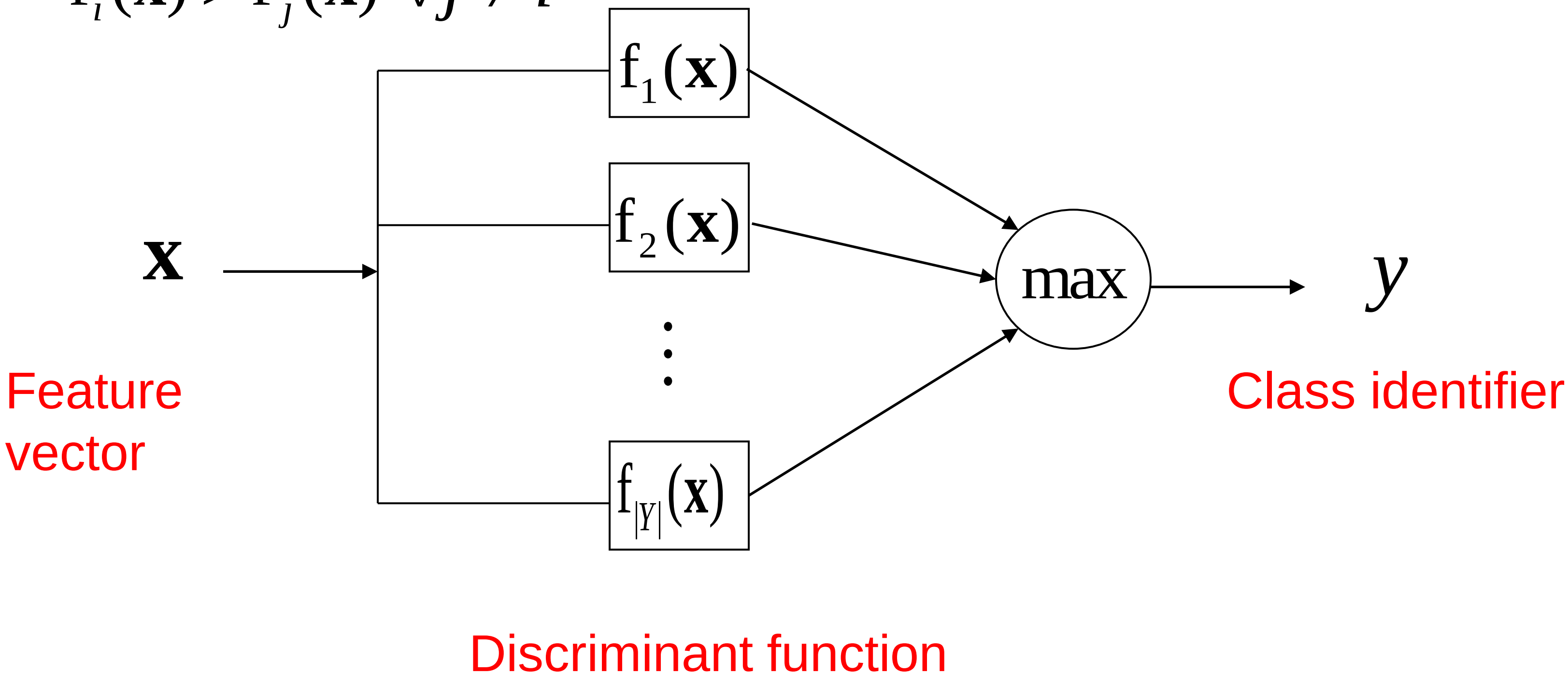
Borders between **decision boundaries** are called decision regions.

Representation of classifier

A classifier is typically represented as a set of discriminant functions

$$\mathrm{f}_i(\mathbf{x}): X \rightarrow \Re, i = 1, \ldots, |Y|$$

The classifier assigns a feature vector **x** to the *i*-the class if $\mathrm{f}_i(\mathbf{x}) > \mathrm{f}_j(\mathbf{x}) \ \forall j \neq i$



Review: What We've Learned So Far

- Bayesian Decision Theory
- Maximum-Likelihood & Bayesian Parameter Estimation
- Parametric Density Estimation
- Nonparametric Density Estimation
  - Parzen-Window, k_n-Nearest-Neighbor

- K-Nearest Neighbor Classifier
- Decision Tree Classifier

Today: Support Vector Machine (SVM)

- A classifier derived from statistical learning theory by Vapnik, et al. in 1992
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task
- Currently, SVM is widely used in object detection & recognition, content-based image retrieval, text recognition, biometrics, speech recognition, etc.
- Also used for regression (will not cover today)

V. Vapnik

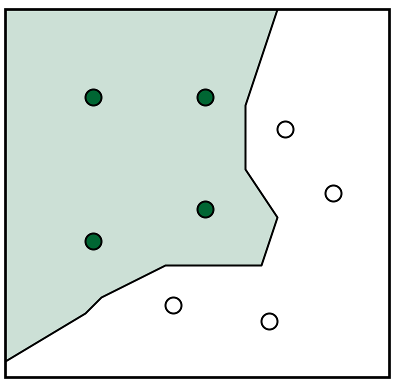
Outline

- Linear Discriminant Function
- Large Margin Linear Classifier
- Nonlinear SVM: The Kernel Trick
- Demo of SVM

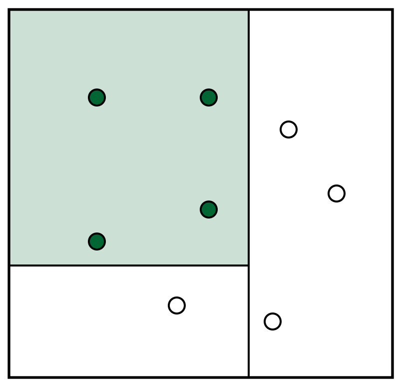
Slides from Jinwei Gu

Discriminant Function

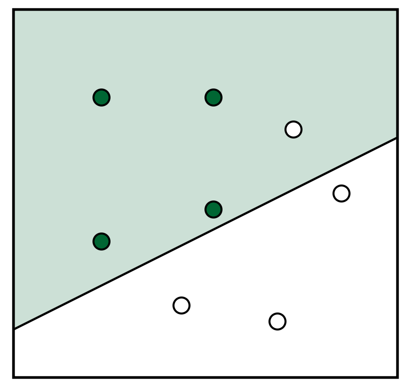
- It can be arbitrary functions of *x*, such as:



Nearest Neighbor

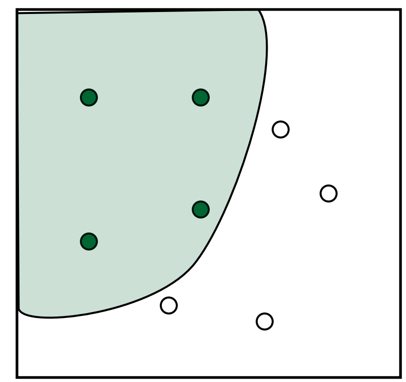


Decision Tree



Linear Functions

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$



Nonlinear Functions

Linear Discriminant Function

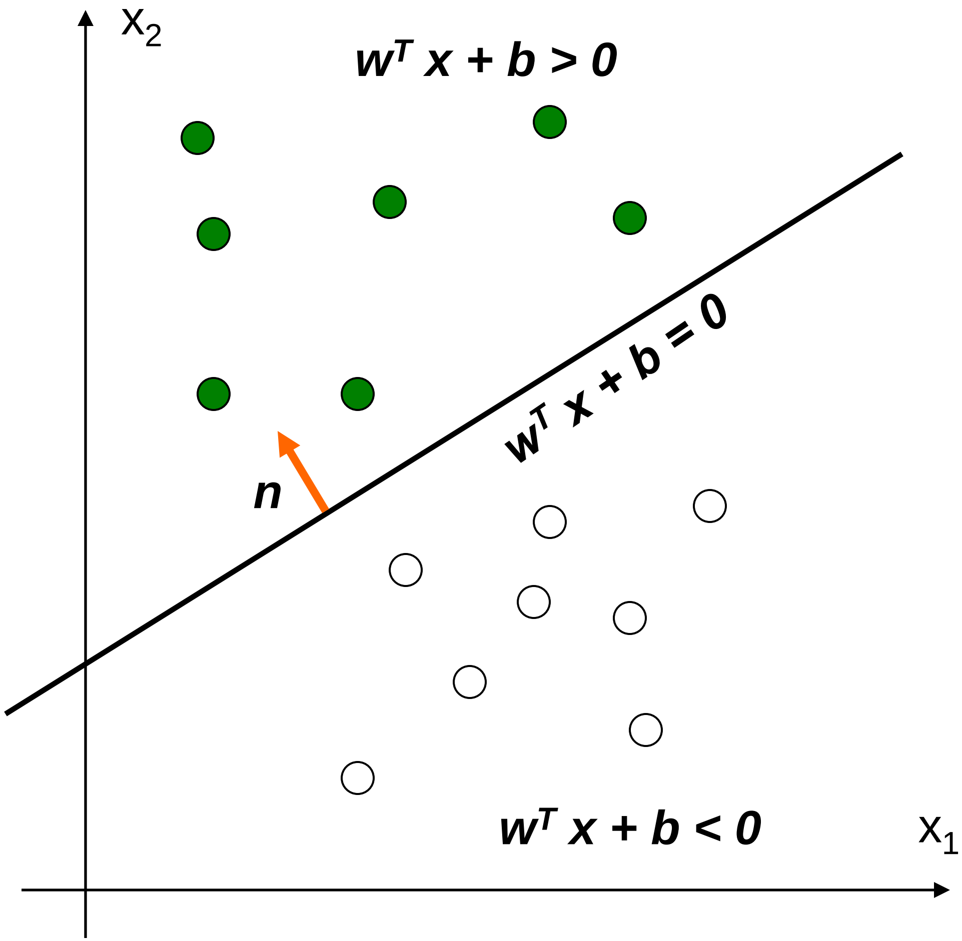
- g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- A hyper-plane in the feature space

- (Unit-length) normal vector of the hyper-plane:

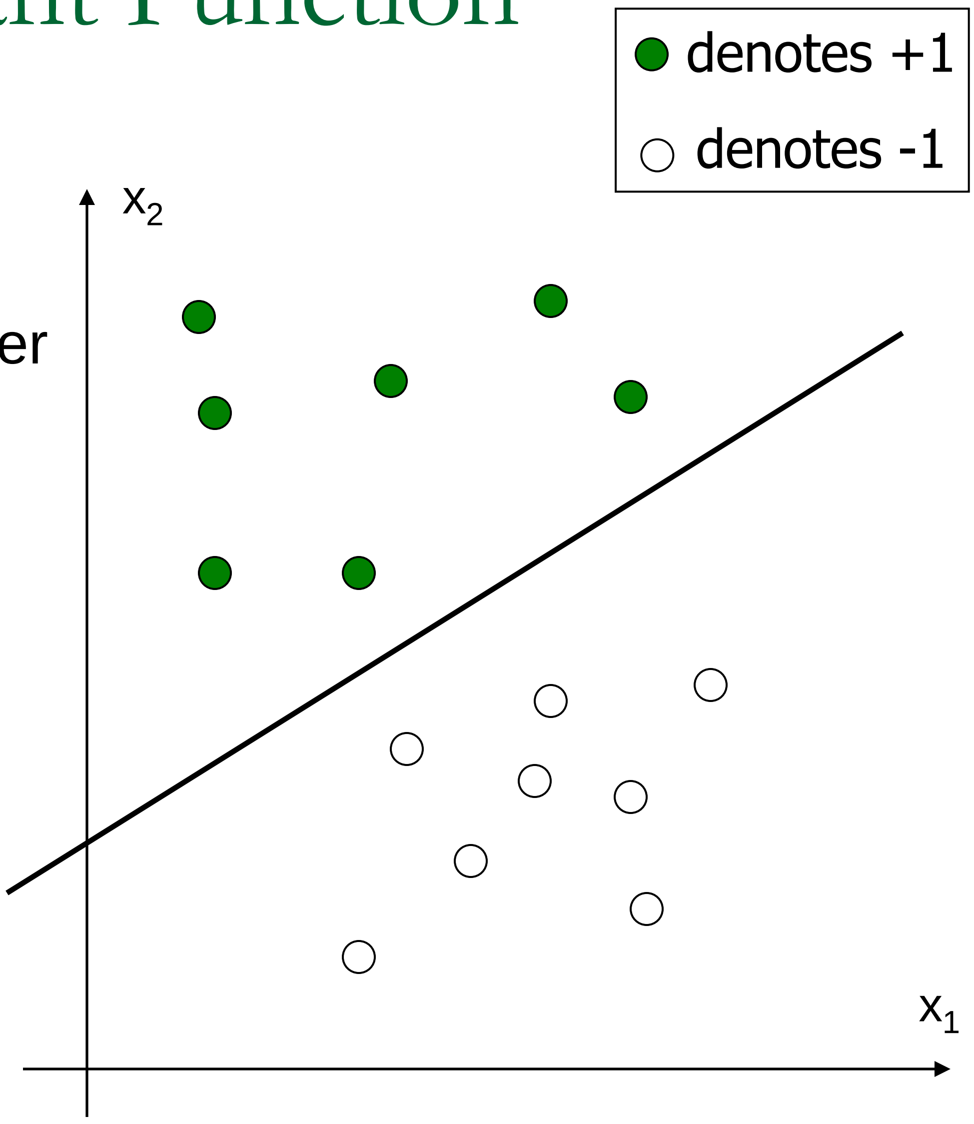
$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

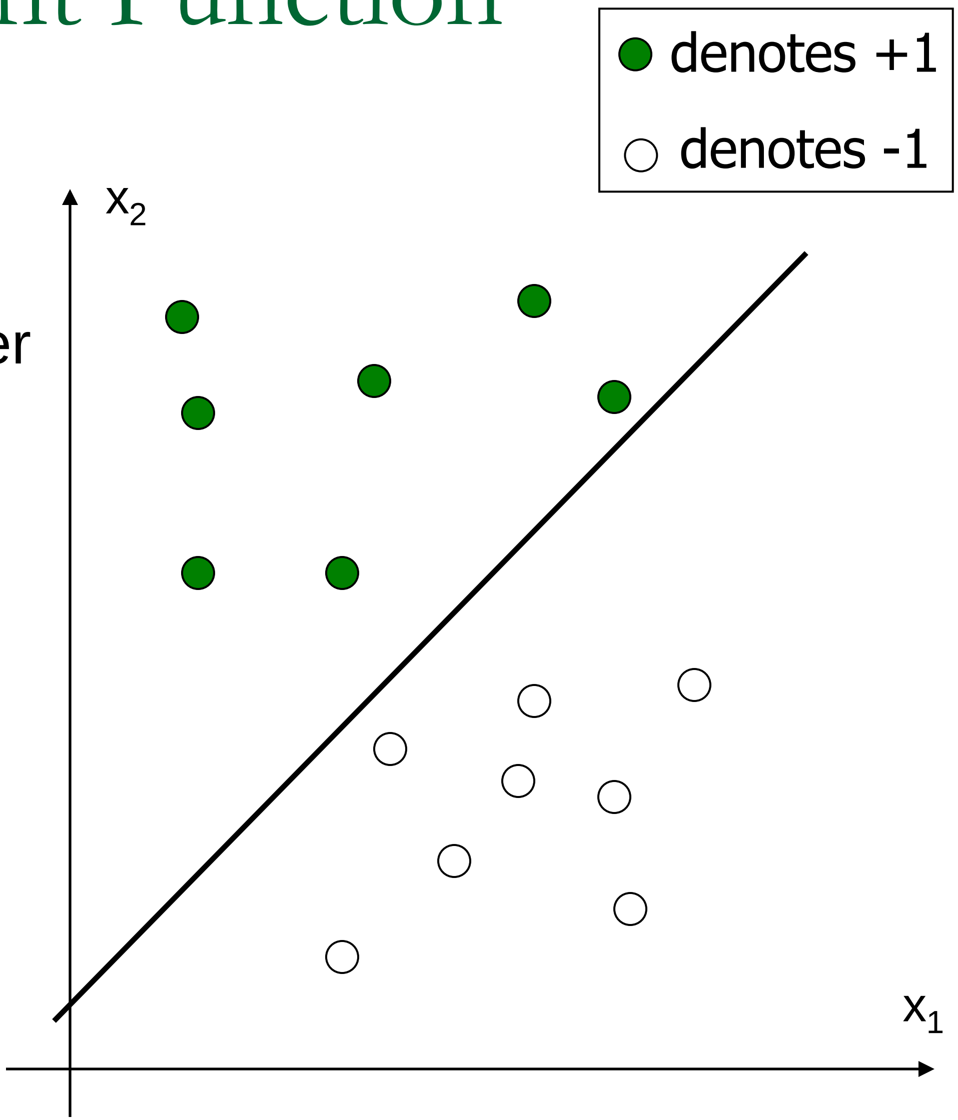
- Infinite number of answers!



Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?

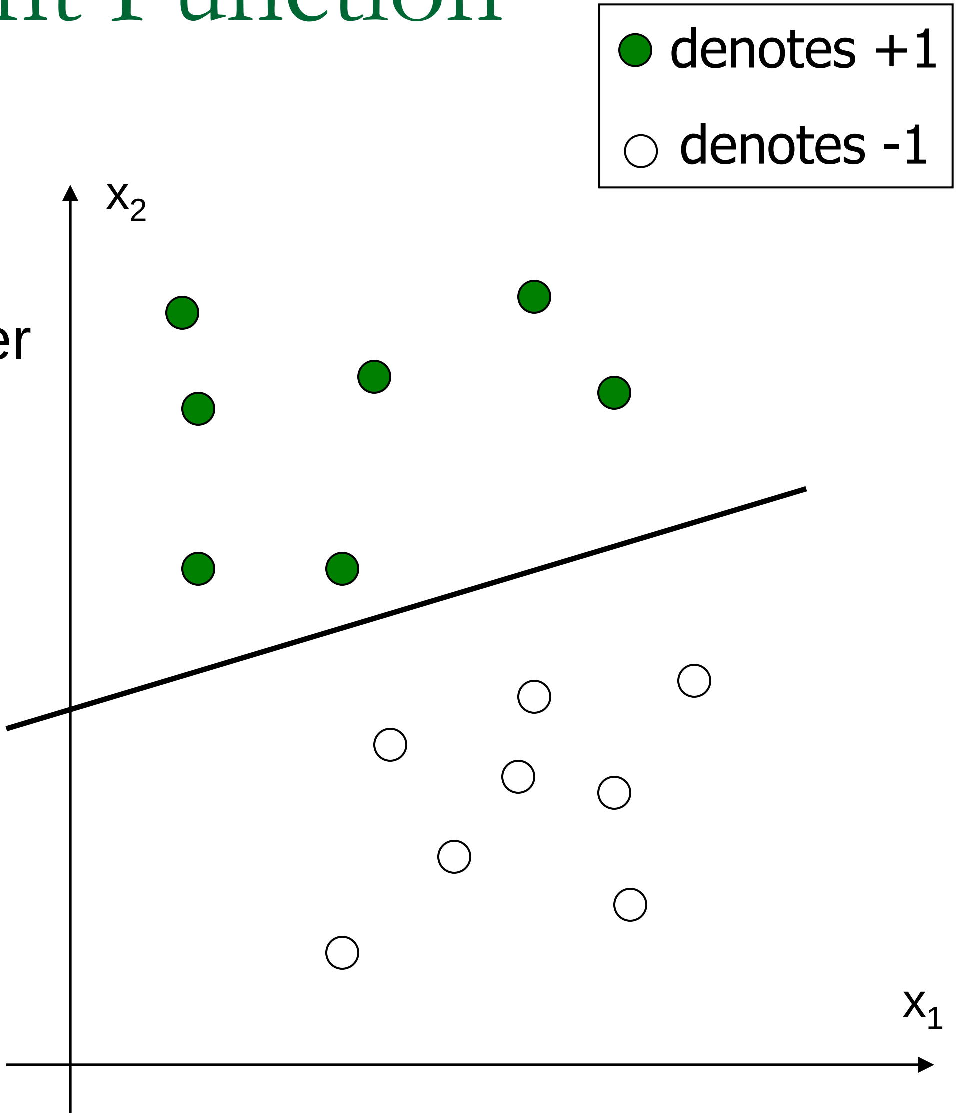
- Infinite number of answers!



Linear Discriminant Function

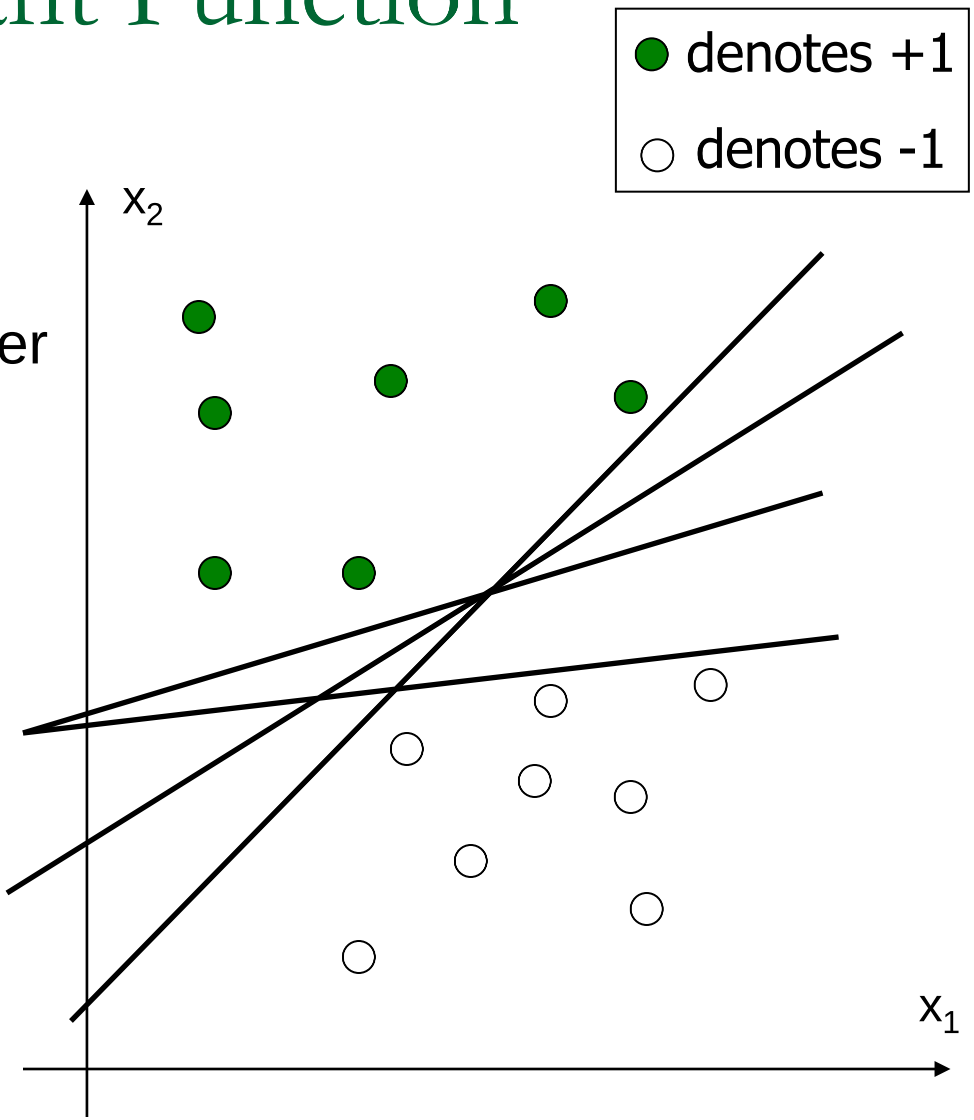
- How would you classify these points using a linear discriminant function in order to minimize the error rate?

- Infinite number of answers!



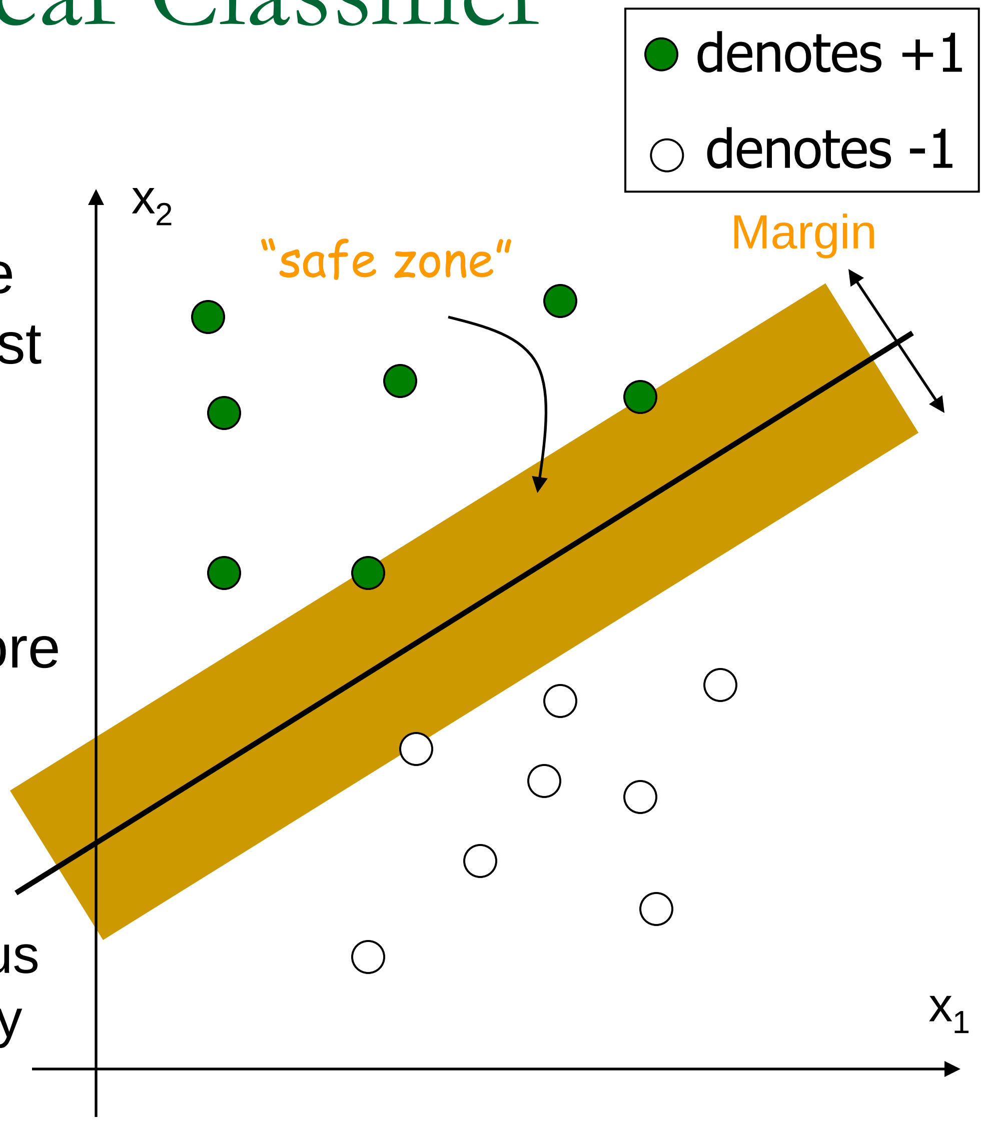
Linear Discriminant Function

- How would you classify these points using a linear discriminant function in order to minimize the error rate?
- Infinite number of answers!
- Which one is the best?



Large Margin Linear Classifier

- The linear discriminant function (classifier) with the maximum margin is the best
- Margin is defined as the width that the boundary could be increased by before hitting a data point
- Why it is the best?
  - Robust to outliners and thus strong generalization ability



Large Margin Linear Classifier

- Given a set of data points:

$\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n$, where

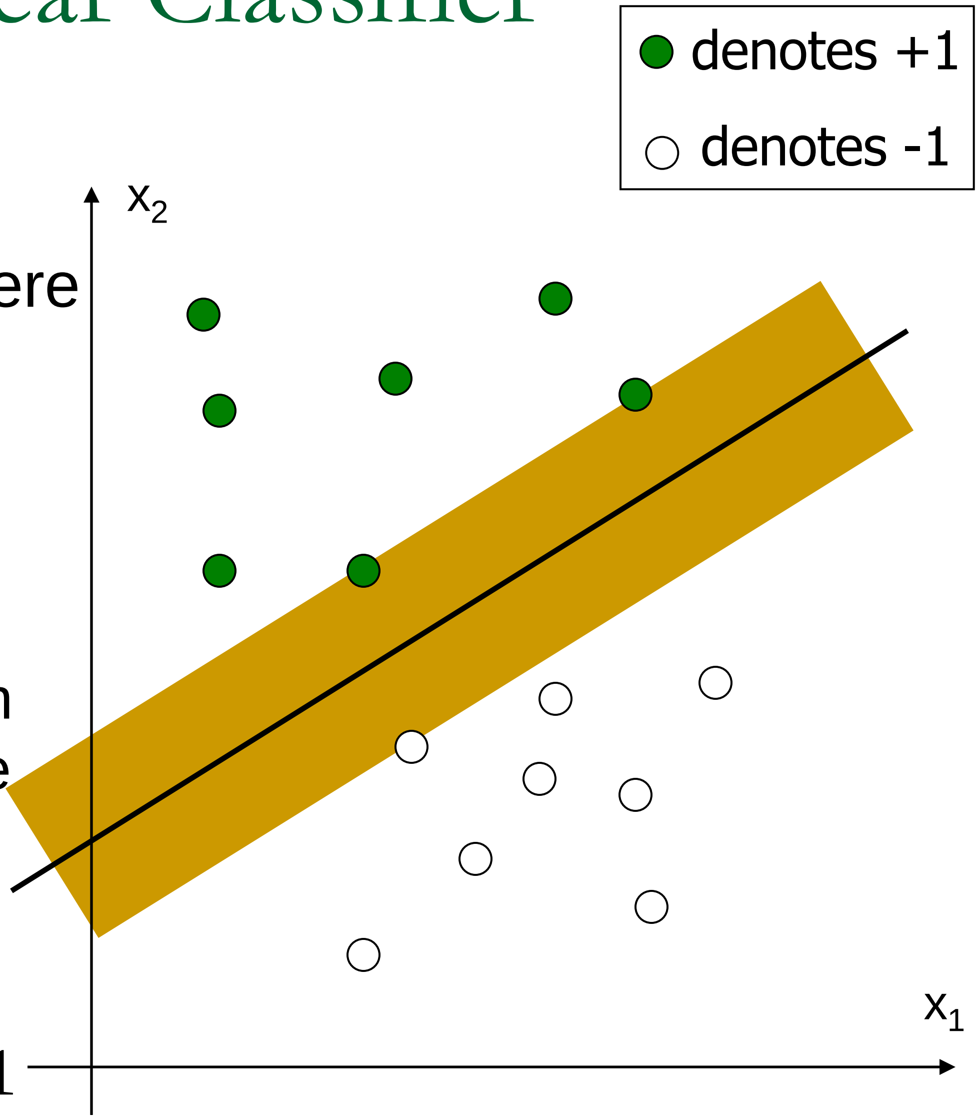
$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b > 0$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b < 0$$

- With a scale transformation on both *w* and *b*, the above is equivalent to

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



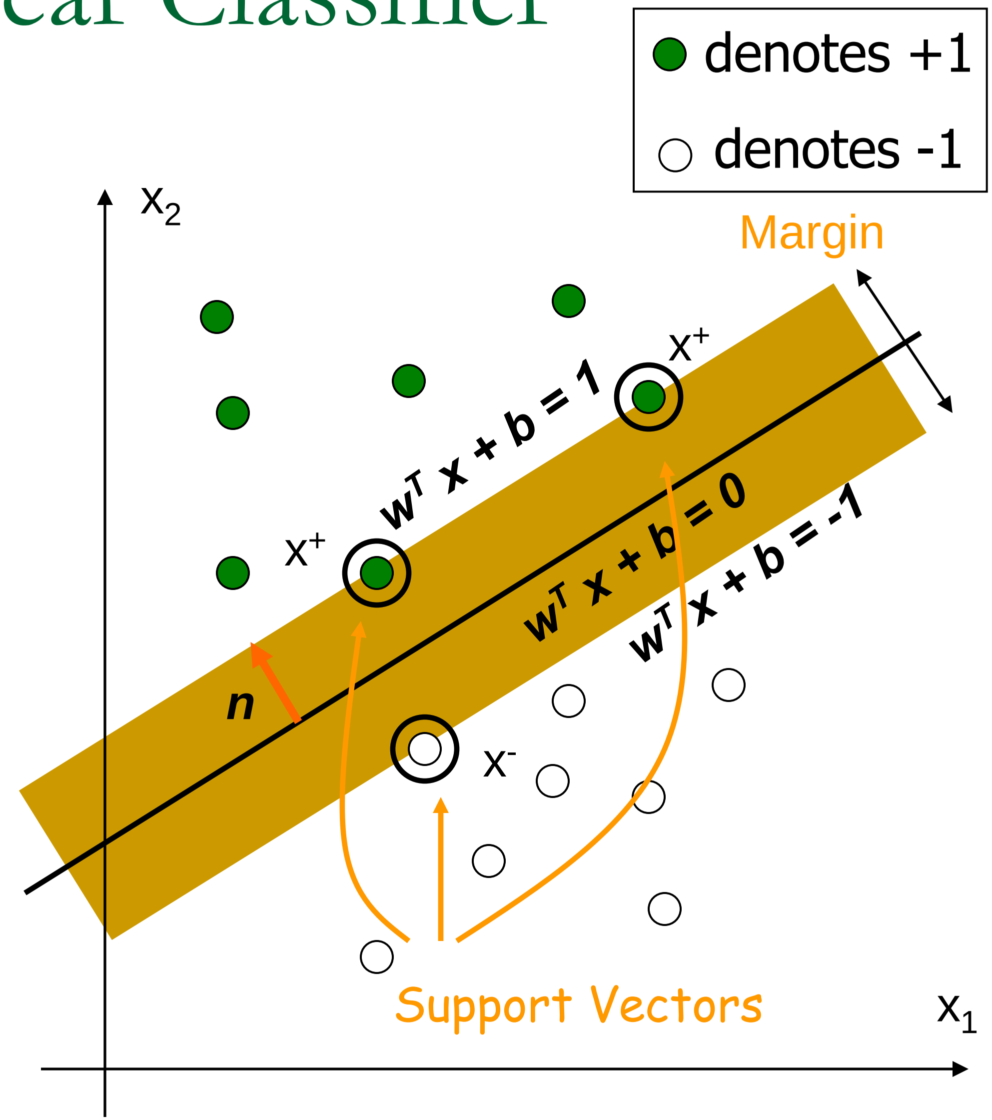
Large Margin Linear Classifier

- We know that

$$\mathbf{w}^T \mathbf{x}^+ + b = 1$$
$$\mathbf{w}^T \mathbf{x}^- + b = -1$$

- The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \mathbf{n}$$
$$= (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$



Large Margin Linear Classifier

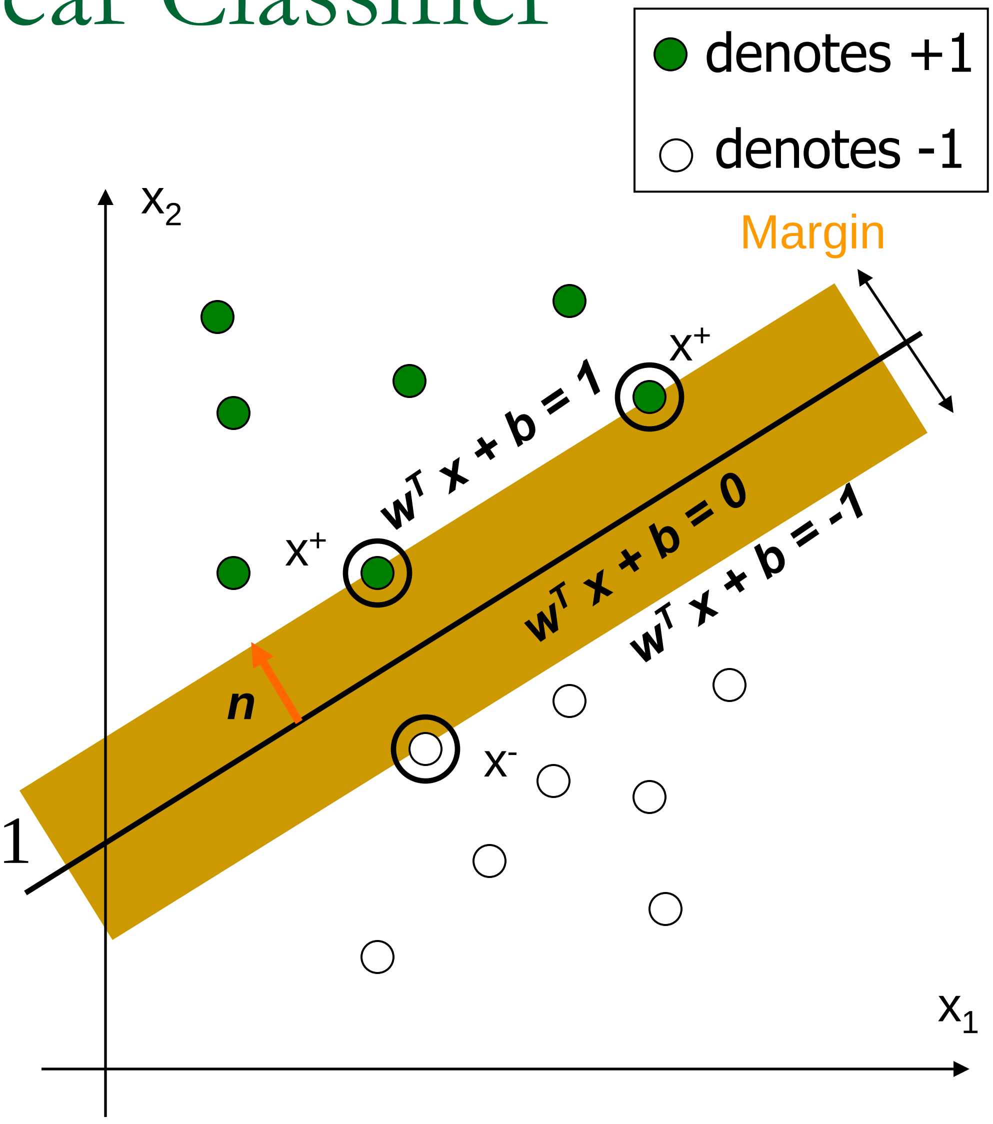
- Formulation:

$$\text{maximize} \quad \frac{2}{\|\mathbf{w}\|}$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T \mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T \mathbf{x}_i + b \leq -1$$



Large Margin Linear Classifier

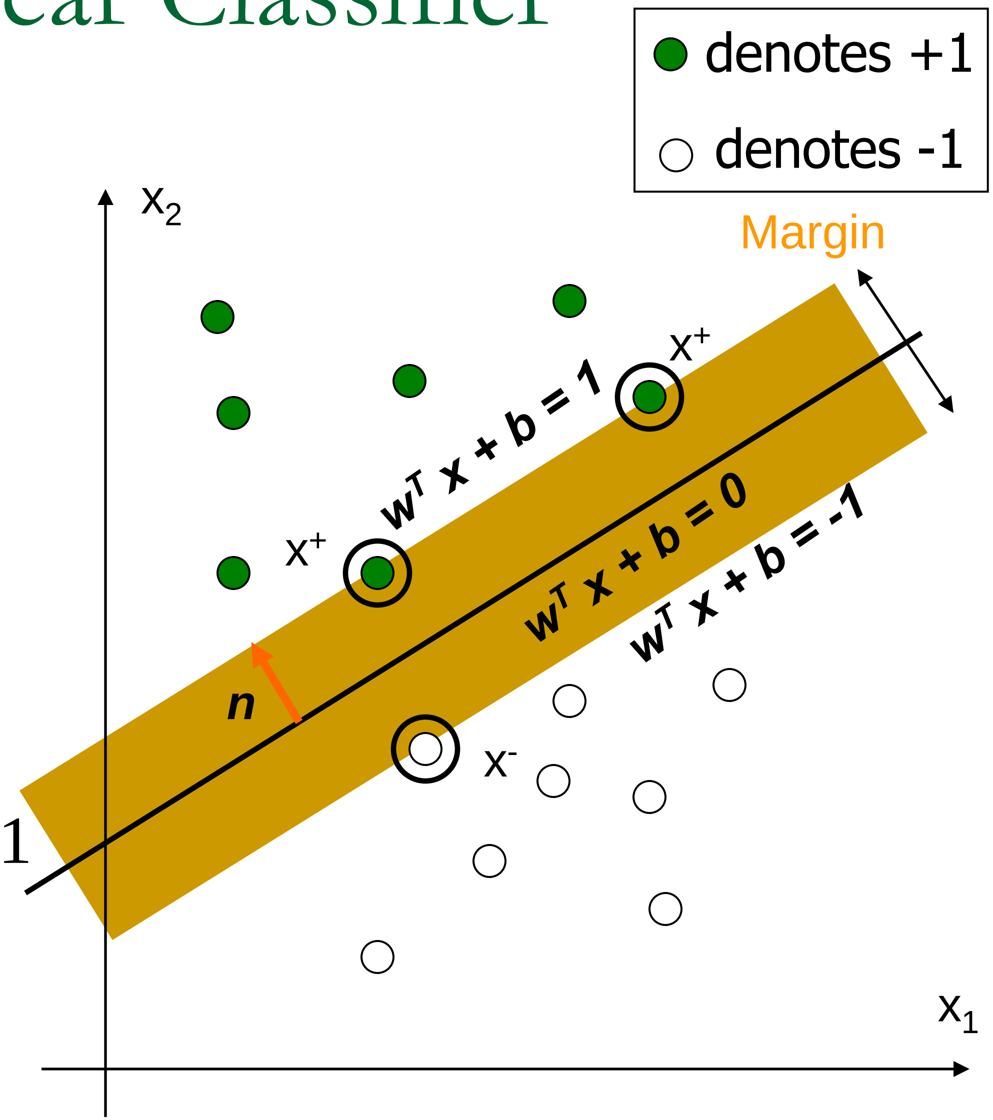
- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$\text{For } y_i = +1, \quad \mathbf{w}^T\mathbf{x}_i + b \geq 1$$

$$\text{For } y_i = -1, \quad \mathbf{w}^T\mathbf{x}_i + b \leq -1$$



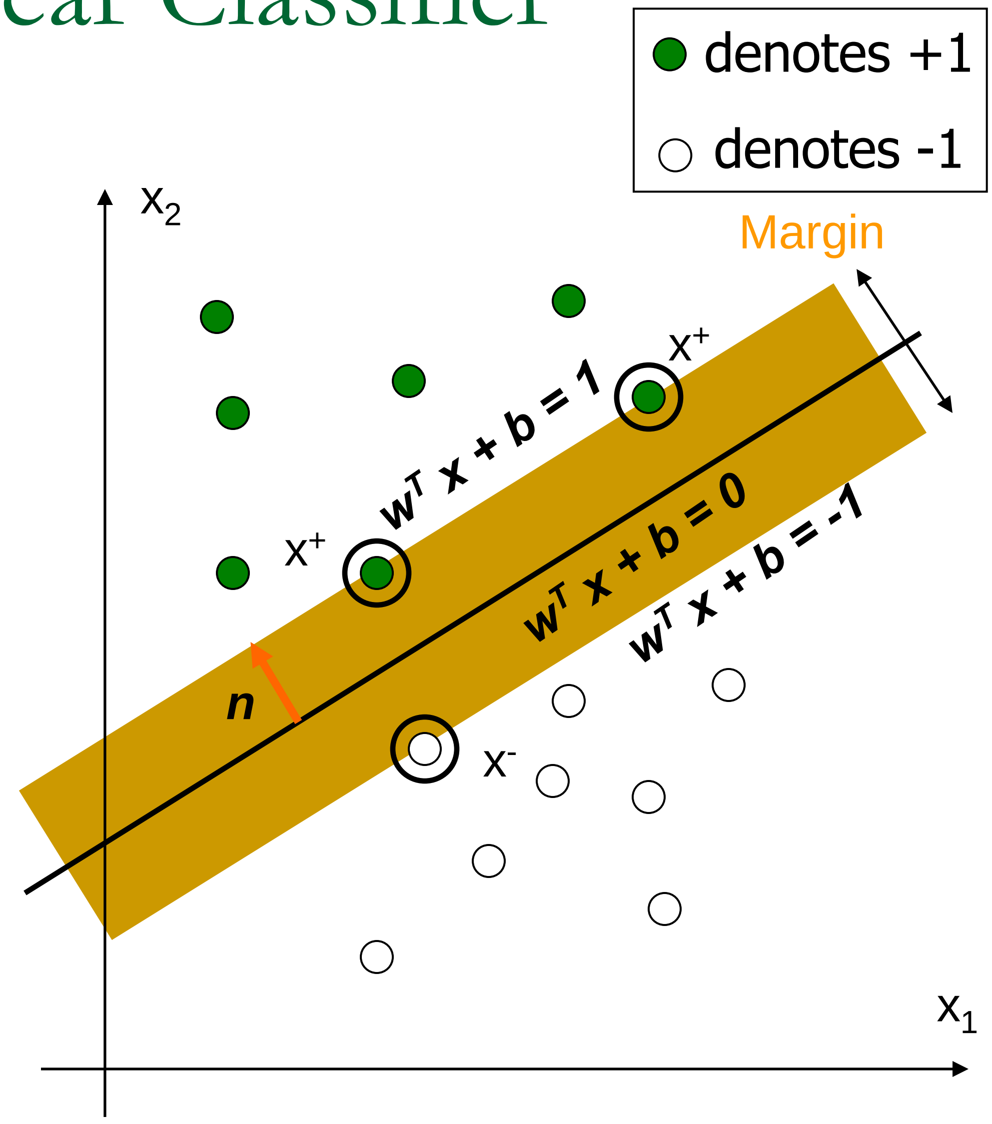
Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



Solving the Optimization Problem

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Lagrangian Function

$$\text{minimize} \quad L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

Solving the Optimization Problem

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \alpha_i \geq 0$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Solving the Optimization Problem

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

s.t. $\alpha_i \geq 0$

Lagrangian Dual Problem

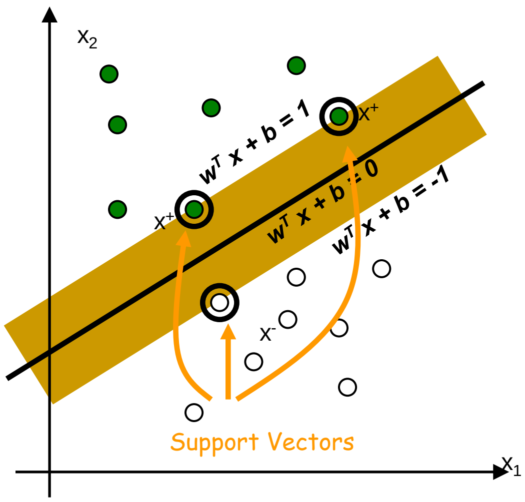
$$\text{maximize } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

s.t. $\alpha_i \geq 0$, and $\sum_{i=1}^{n} \alpha_i y_i = 0$

Solving the Optimization Problem

- From KKT condition, we know:

$$\alpha_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right) = 0$$



- Thus, only support vectors have $\alpha_i \neq 0$

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \sum_{i \in \mathrm{SV}} \alpha_i y_i \mathbf{x}_i$$

get b from $y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0,$

where $\mathbf{x}_i$ is support vector

Solving the Optimization Problem

- The linear discriminant function is:

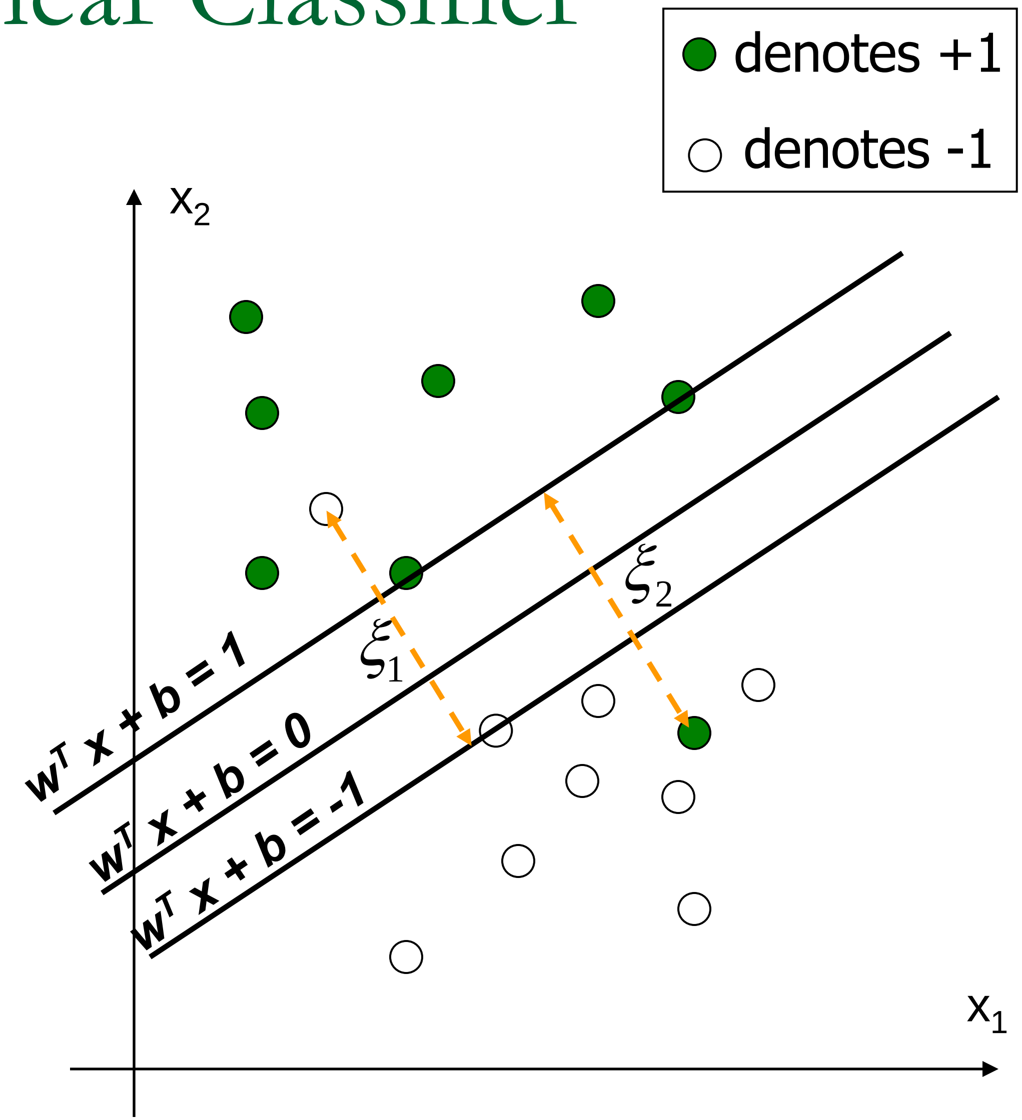
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i \in \mathrm{SV}} \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice it relies on a *dot product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$

- Also keep in mind that solving the optimization problem involved computing the dot products $\boldsymbol{x}_i^T \boldsymbol{x}_j$ between all pairs of training points

Large Margin Linear Classifier

- What if data is not linear separable? (noisy data, outliers, etc.)
- Slack variables ξ_i can be added to allow mis-classification of difficult or noisy data points



Large Margin Linear Classifier

- Formulation:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- Parameter *C* can be viewed as a way to control over-fitting.

Large Margin Linear Classifier

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

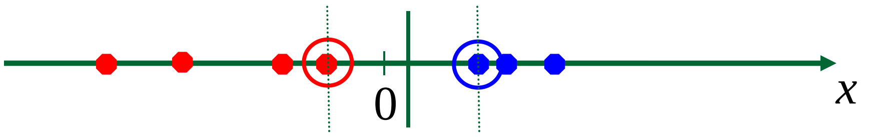
such that

$$0 \le \alpha_i \le C$$

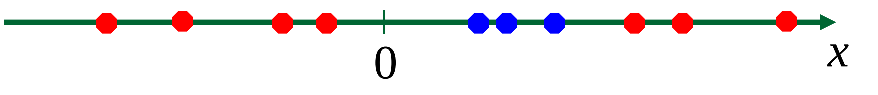
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Non-linear SVMs

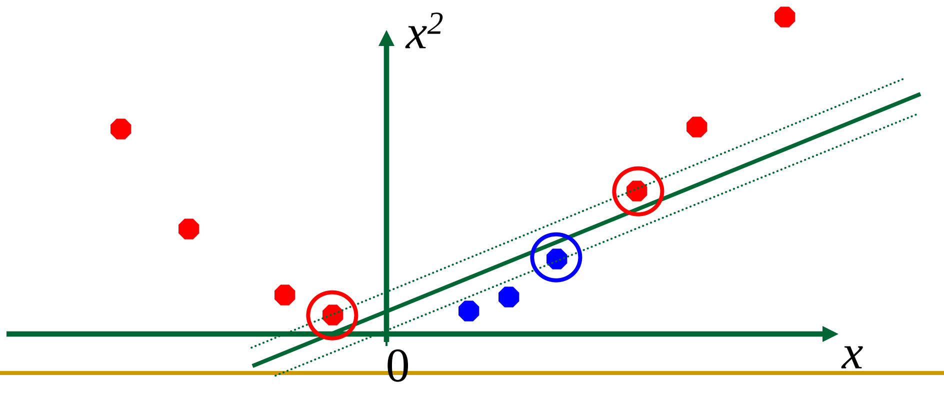
- Datasets that are linearly separable with noise work out great:



- But what are we going to do if the dataset is just too hard?



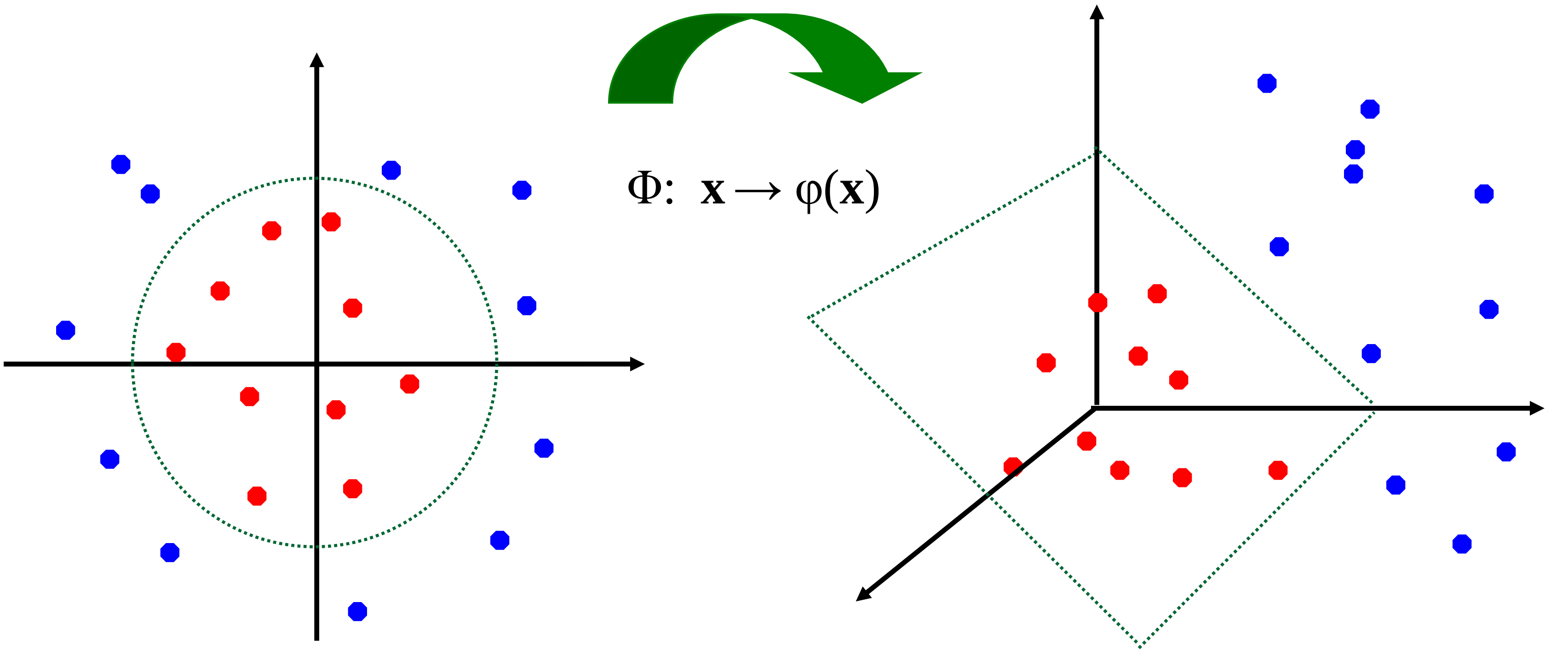
- How about… mapping data to a higher-dimensional space:



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Non-linear SVMs: Feature Space

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



Nonlinear SVMs: The Kernel Trick

- With this mapping, our discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b = \sum_{i \in \mathrm{SV}} \alpha_i \boxed{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b$$

- No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Nonlinear SVMs: The Kernel Trick

- An example:

2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$;

let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

$$
\begin{aligned}
K(\mathbf{x}_i,\mathbf{x}_j)&=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2, \\
&= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}] \\
&= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j), \quad \text{where } \varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]
\end{aligned}
$$

Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

  - Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

- In general, functions that satisfy *Mercer's condition* can be kernel functions.

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \mathrm{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same.

Support Vector Machine: Algorithm

- 1. Choose a kernel function
- 2. Choose a value for *C*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

Some Issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures

- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- Optimization criterion – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Summary: Support Vector Machine

- 1. Large Margin Classifier
  - Better generalization ability & less over-fitting

- 2. The Kernel Trick
  - Map data points to higher dimensional space in order to make them linearly separable.
  - Since only dot product is used, we do not need to represent the mapping explicitly.

Additional Resource

- http://www.kernel-machines.org/
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Multiclass classification

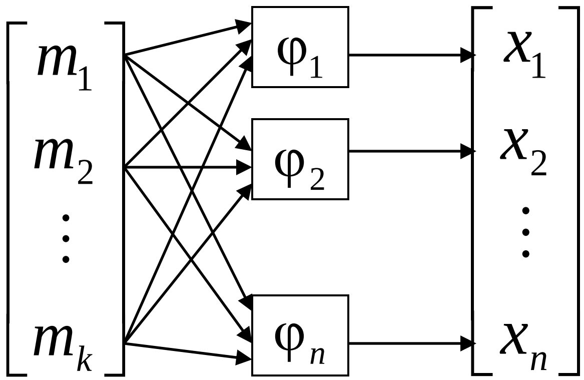
- Reduction techniques
  - Conventional approaches
    - One-against-All
      - K two-class problems
    - Pairwise
      - K(K - 1)/2 two-class problems
    - Decision-Tree-Based
    - DAG (Directed Acyclic Graph)
    - Error-Correcting Output Codes

Multiclass classification

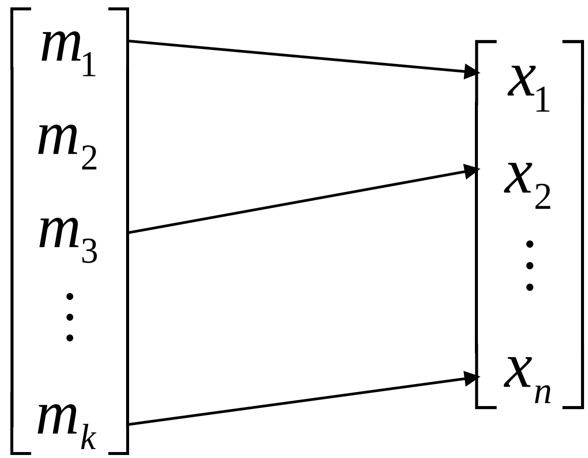
- Reduction techniques
  - Conventional approaches
    - apply binary classifier 1 to test example and get prediction F1 (0/1)
    - apply binary classifier 2 to test example and get prediction F2 (0/1)
    - …
    - apply binary classifier M to test example and get prediction FM (0/1)
    - use all M classifications to get the final multiclass classification 1..K

Feature extraction methods

Feature extraction



Feature selection



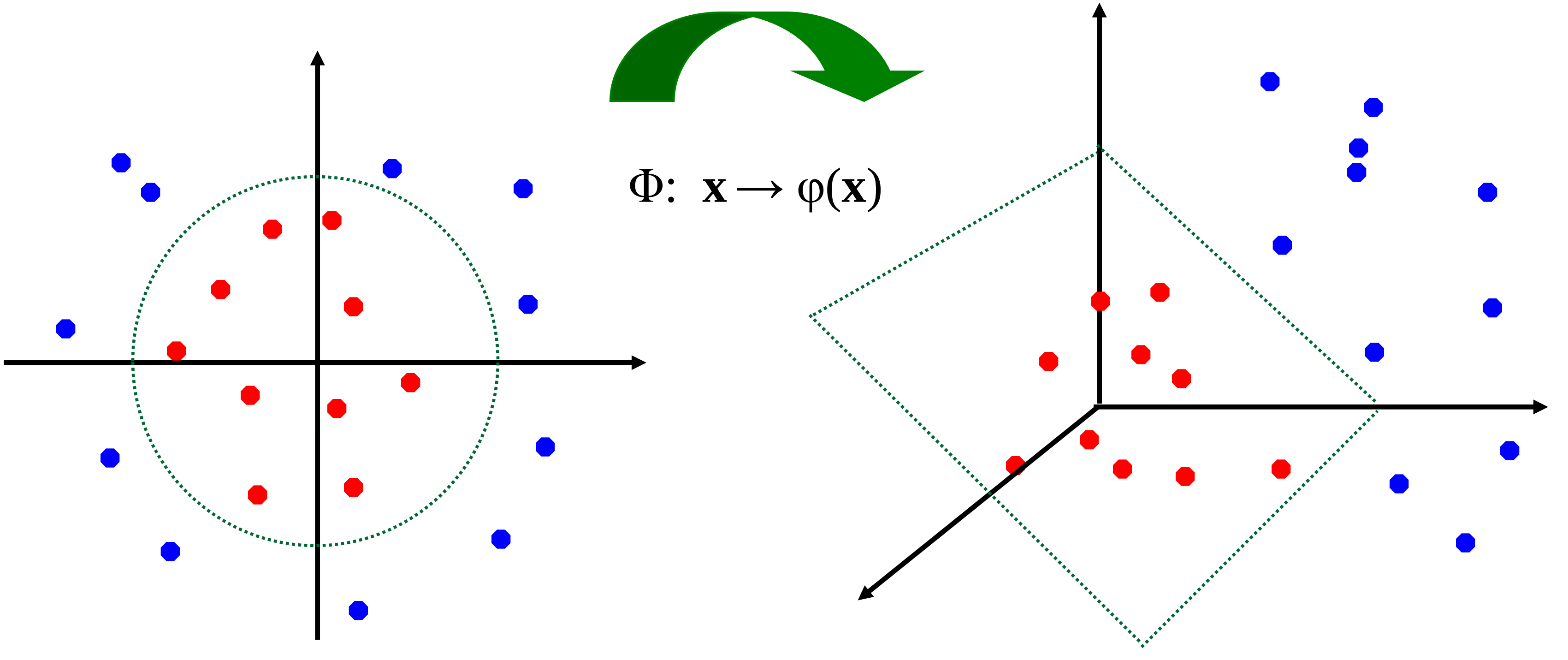
Problem can be expressed as optimization of parameters of feature extractor $\varphi(\theta)$

Supervised methods: objective function is a criterion of separability (discriminability) of labeled examples, e.g., linear discriminat analysis (LDA).

Unsupervised methods: lower dimensional representation which preserves important characteristics of input data is sought for, e.g., principal component analysis (PCA).

Non-linear SVMs: Feature Space

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Artificial Neural Networks

Slides from Andrew L. Nelson and Torsten Reil

What are Neural Networks?

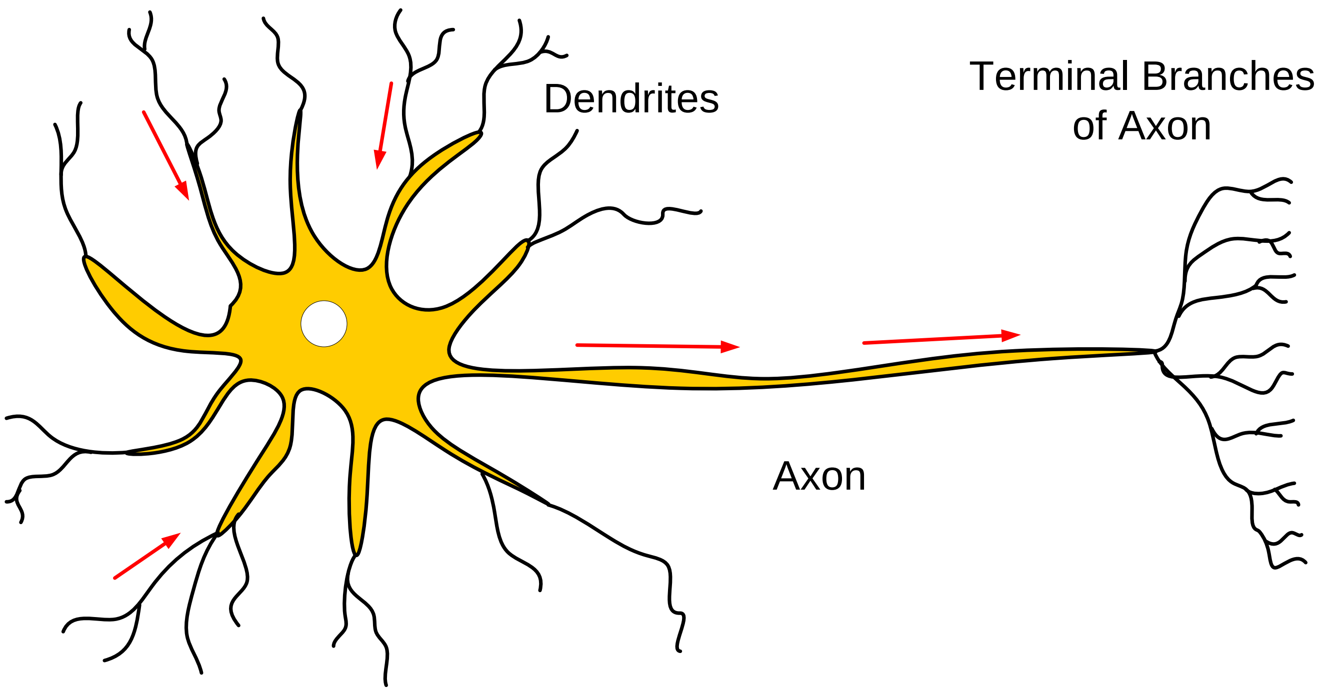
- Models of the brain and nervous system
- Highly parallel
  - Process information much more like the brain than a serial computer
- Learning

- Very simple principles
- Very complex behaviours

- Applications
  - As powerful problem solvers
  - As biological models

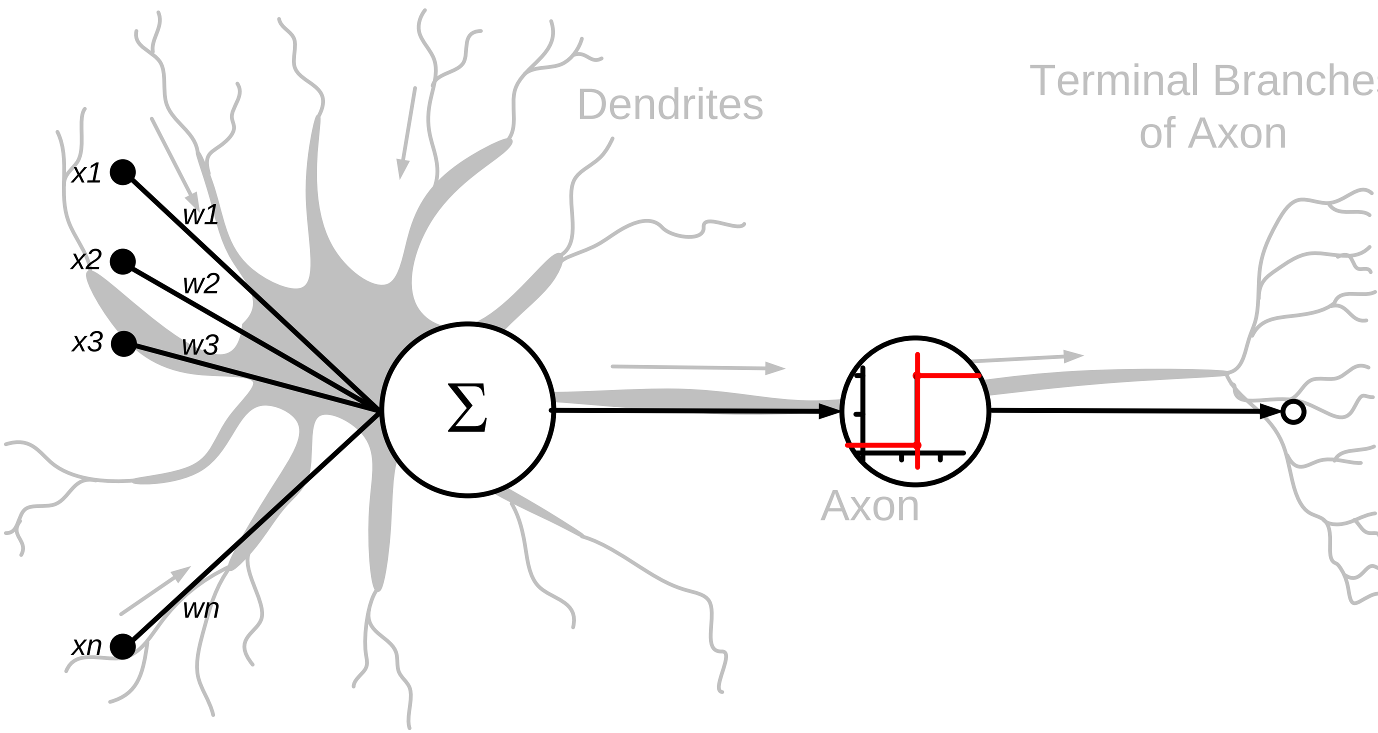
Biologically Inspired

- Electro-chemical signals
- Threshold output firing



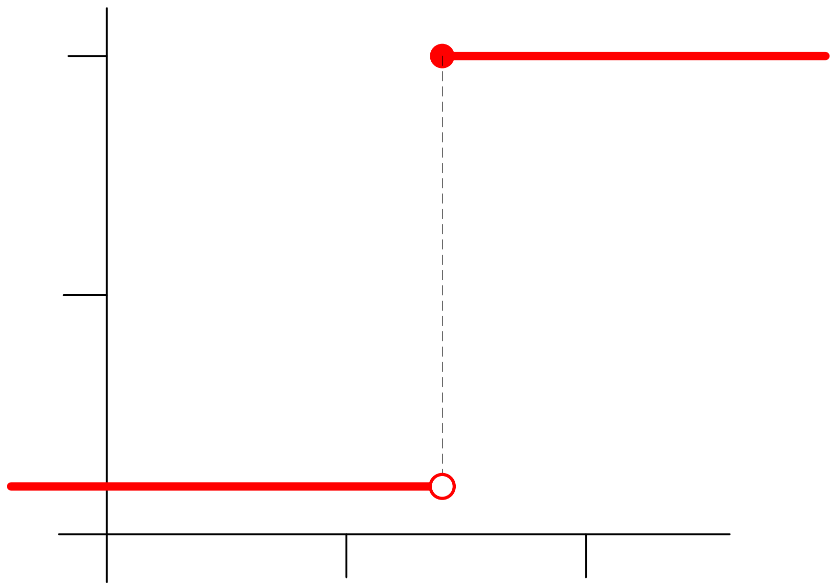
The Perceptron

- Binary classifier functions
- Threshold activation function



The Perceptron: Threshold Activation Function

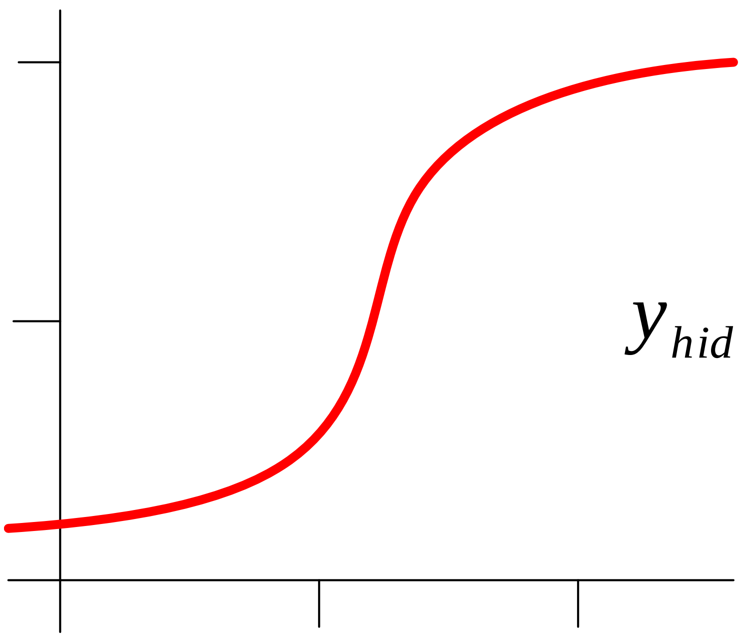
- Binary classifier functions
- Threshold activation function



Step Threshold

Nonlinear Activation Functions

- Sigmoid Neuron unit function

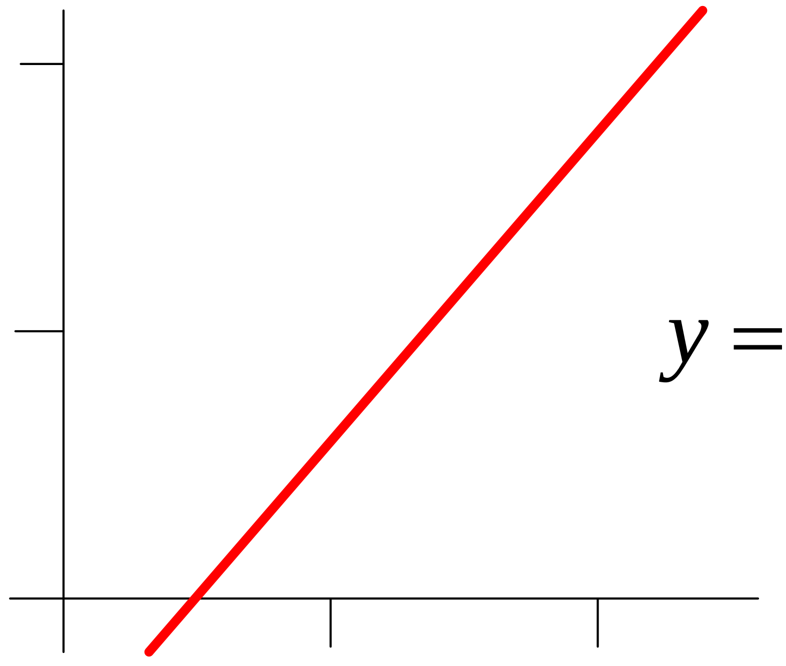


Sigmoid

$$y_{hid}(u) = \frac{1}{1+e^{-u}}$$

Linear Activation functions

- Output is scaled sum of inputs

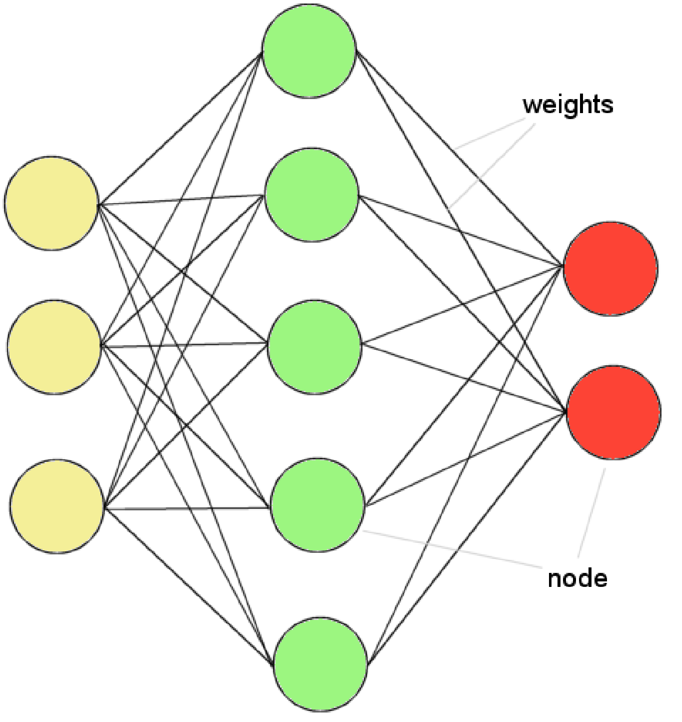


$$y = u = \sum_{n=1}^{N} w_n x_n$$

ANNs – The basics

- ANNs incorporate the two fundamental components of biological neural nets:

1. Neurones (nodes)
2. Synapses (weights)



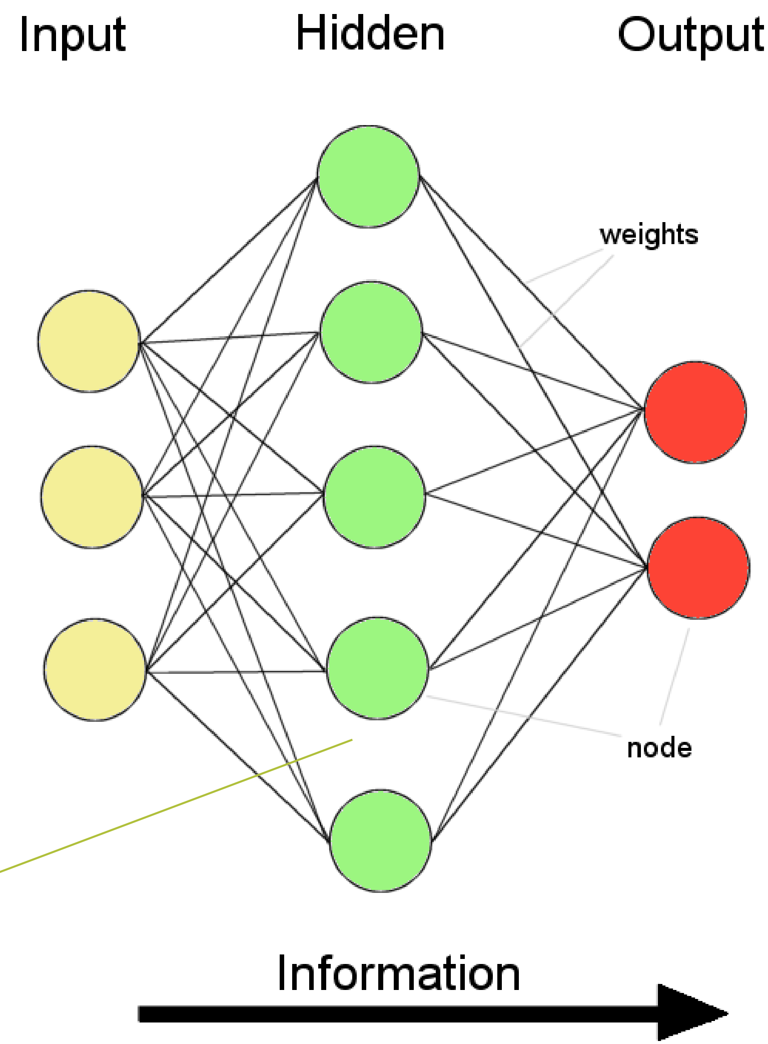
Feed-forward nets

Information flow is unidirectional

- Data is presented to *Input layer*
- Passed on to *Hidden Layer*
- Passed on to *Output layer*

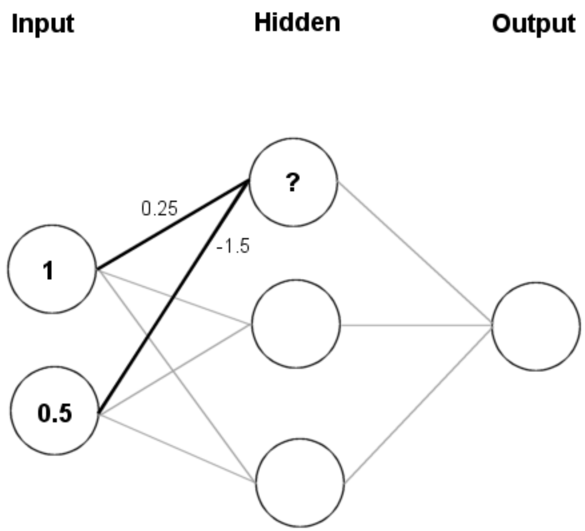
Information is distributed

Information processing is parallel



Internal representation (interpretation) of data

- Feeding data through the net:



$(1 \times 0.25) + (0.5 \times (-1.5)) = 0.25 + (-0.75) =$ **- 0.5**

Squashing: $$\frac{1}{1+e^{0.5}} = 0.3775$$

- Data is presented to the network in the form of activations in the input layer

- Examples
  - Pixel intensity (for pictures)
  - Molecule concentrations (for artificial nose)
  - Share prices (for stock market prediction)

- Data usually requires preprocessing
  - Analogous to senses in biology

- How to represent more abstract data, e.g. a name?
  - Choose a pattern, e.g.
    - 0-0-1 for “Chris”
    - 0-1-0 for “Becky”

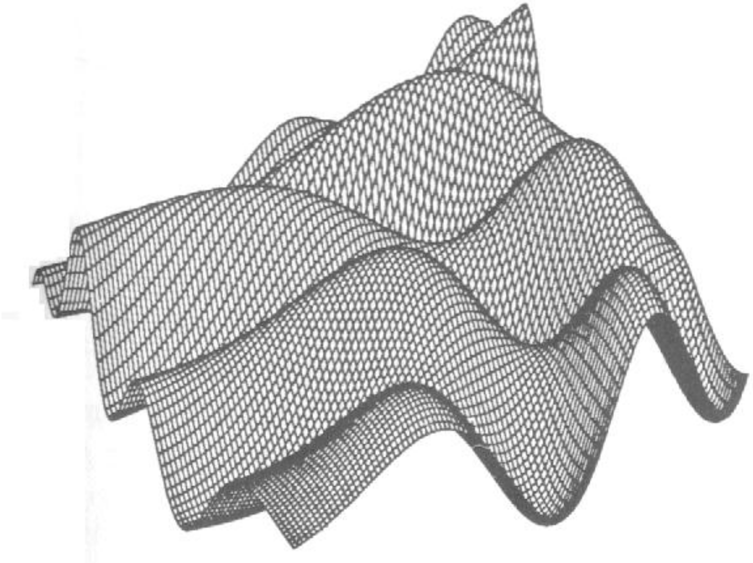
- Weight settings determine the behaviour of a network

→ How can we find the right weights?

Training the Network - Learning

- Backpropagation
  - Requires training set (input / output pairs)
  - Starts with small random weights
  - Error is used to adjust weights (supervised learning)

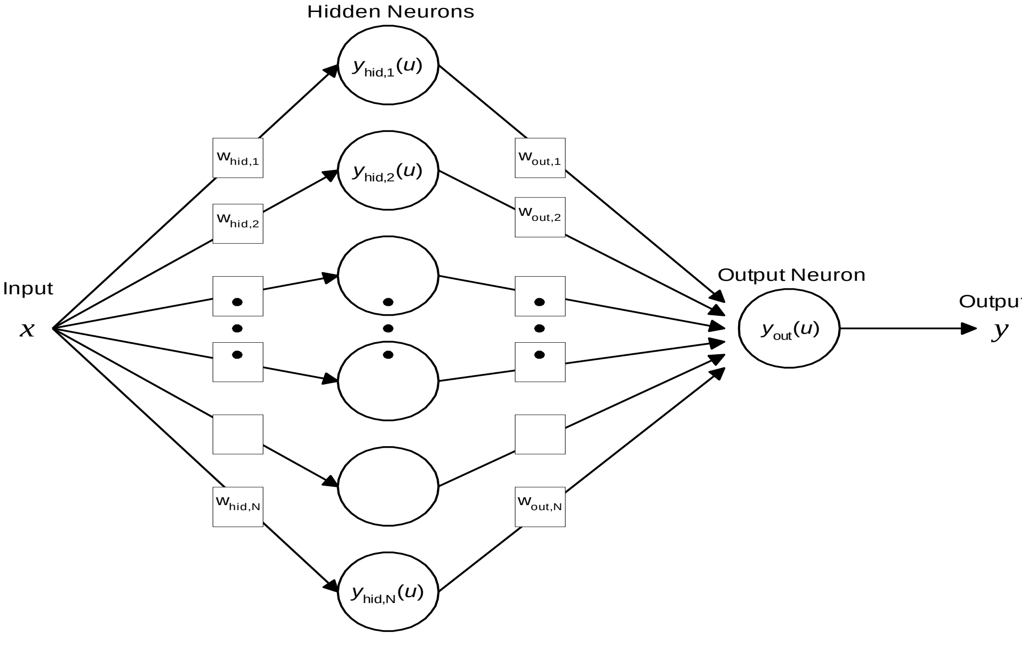
  → Gradient descent on error landscape



Training Data Set

Adjust weights (w) to learn a given target function: $y = f(x)$

Given a set of training data $X \rightarrow Y$



Training Weights: Error Back-Propagation (BP)

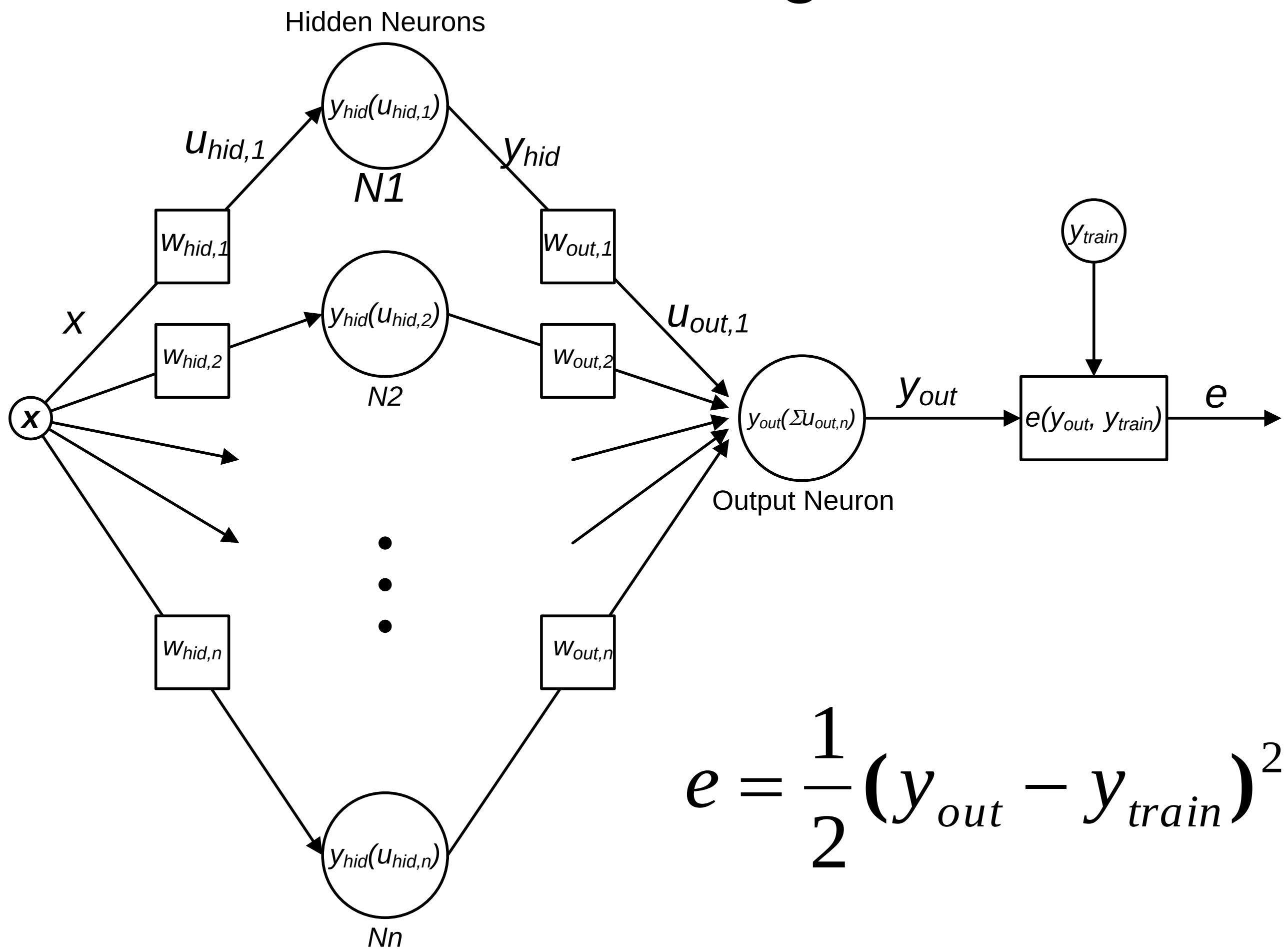
- Weight update formula:

$$w(k+1) = w(k) + \Delta w$$

$$\Delta w(i) = \eta * \frac{\partial e(i)}{\partial w}$$

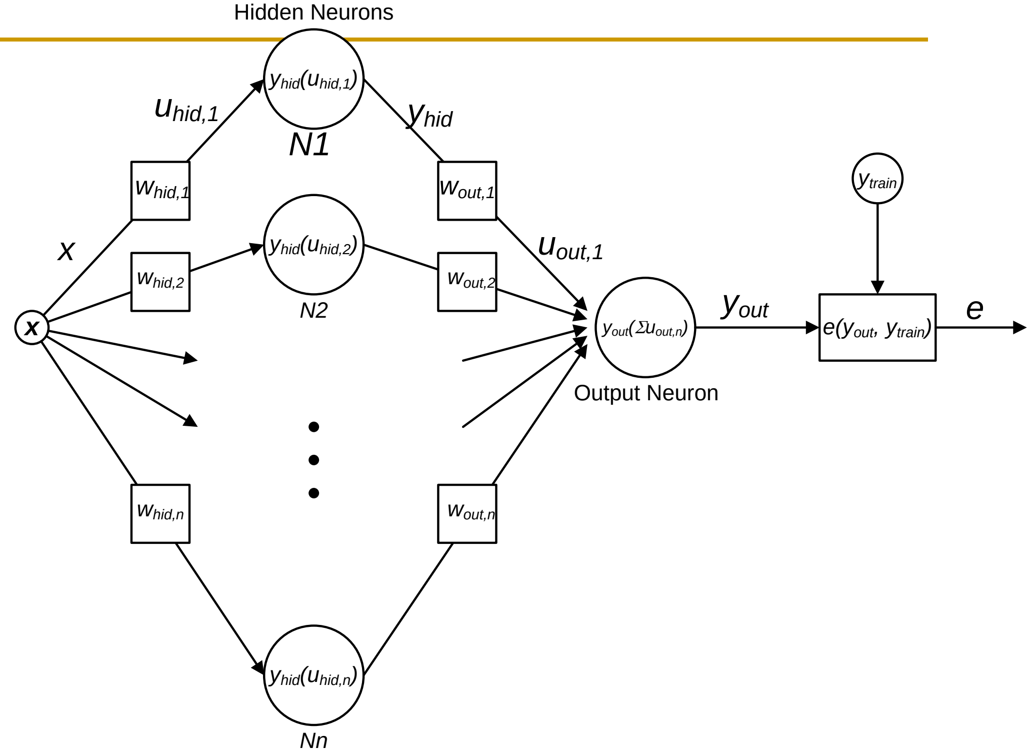
Error Back-Propagation (BP)

Training error term: e



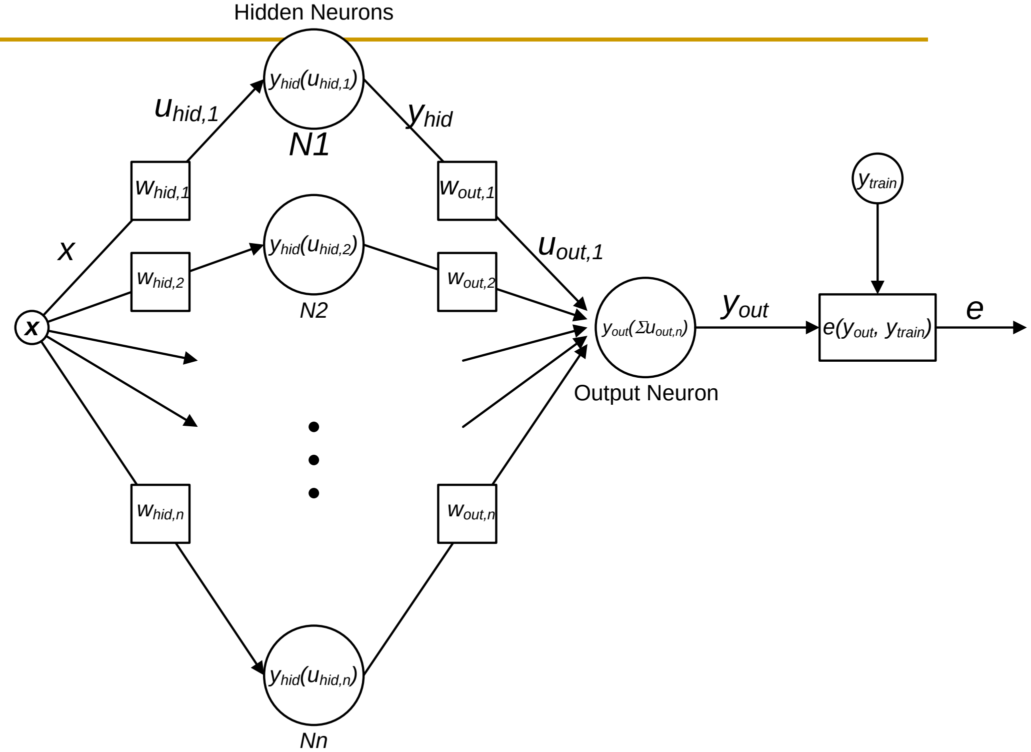
$$e = \frac{1}{2}(y_{out} - y_{train})^2$$

BP Formulation



$$
\begin{aligned}
e(y_{out}, y_{train}) &= e(y_{out}(u_{out,1}), y_{train}) \\
&= e(y_{out}(w_{out,1} y_{hid,1}), y_{train}) \\
&= e(y_{out}(w_{out,1} y_{hid}(u_{hid,1})), y_{train}) \\
&= e(y_{out}(w_{out,1} y_{hid}(w_{hid,1} x)), y_{train})
\end{aligned}
$$

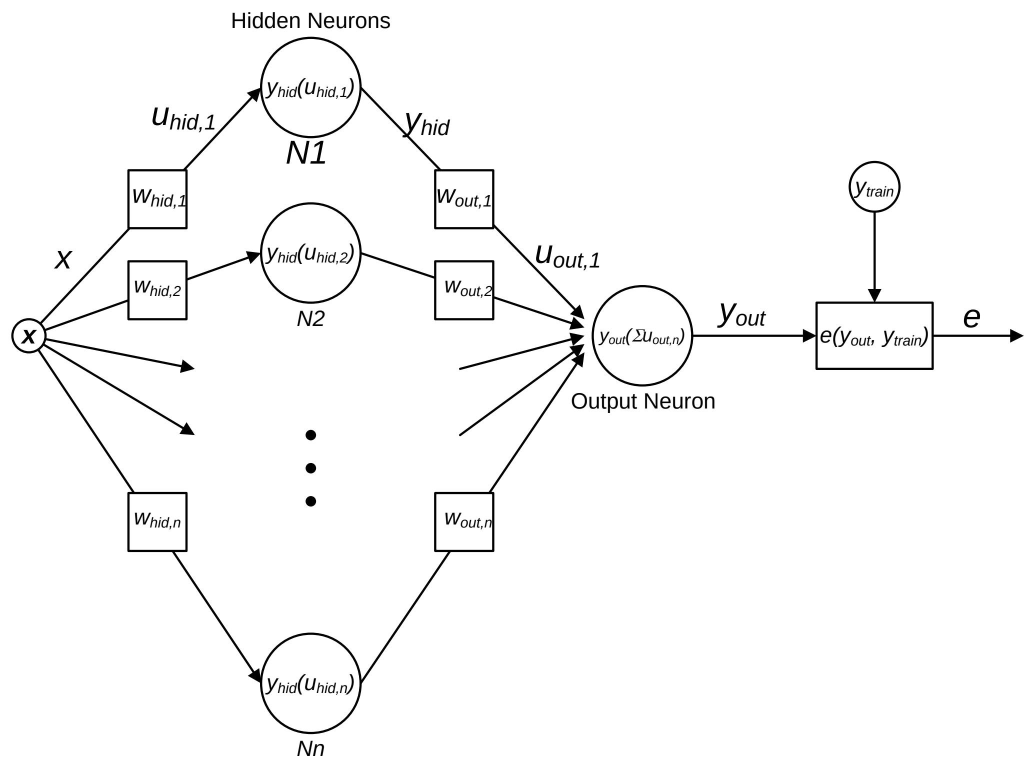
BP Formulation



$$\frac{\partial e}{\partial w_{hid}} = \frac{\partial e}{\partial y_{out}} \frac{\partial y_{out}}{\partial u_{out,1}} \frac{\partial u_{out,1}}{\partial y_{hid}} \frac{\partial y_{hid}}{\partial u_{hid,1}} \frac{\partial u_{hid,1}}{\partial w_{hid,1}}$$

$$\frac{\partial e}{\partial w_{hid}} = \frac{\partial u_{hid,1}}{\partial w_{hid,1}} \frac{\partial y_{hid}}{\partial u_{hid,1}} \frac{\partial u_{out,1}}{\partial y_{hid}} \frac{\partial y_{out}}{\partial u_{out,1}} \frac{\partial e}{\partial y_{out}}$$

BP Formulation

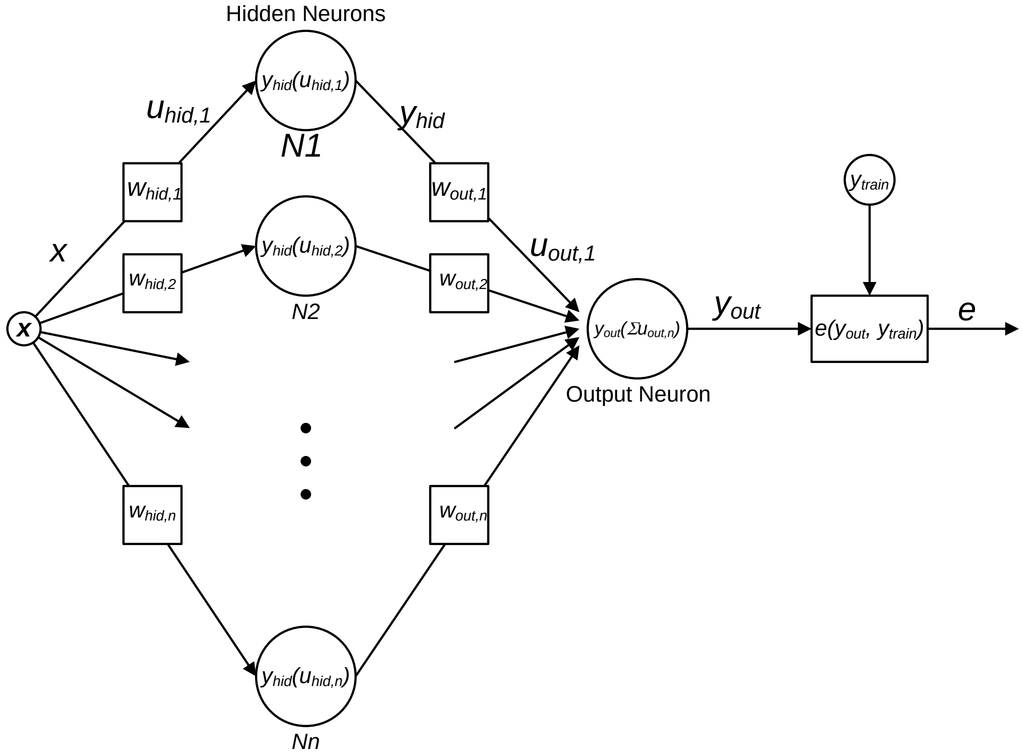


$$\frac{\partial u_{hid,1}}{\partial w_{hid,1}} = \frac{\partial}{\partial w_{hid,1}} w_{hid,1} x$$

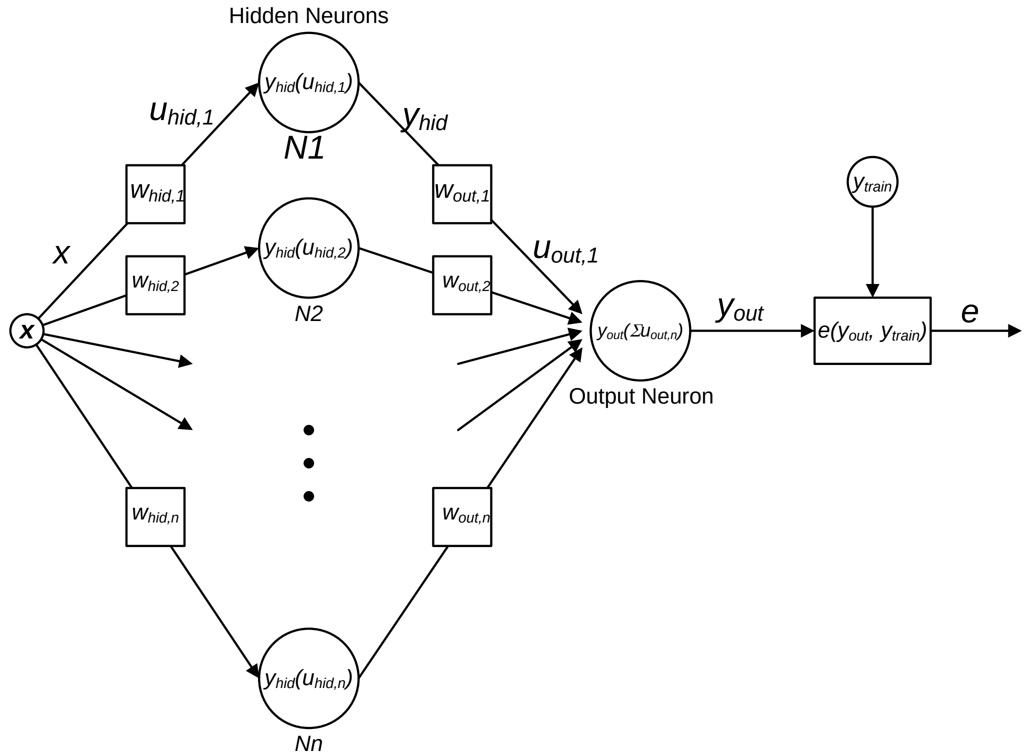
$$= x$$

BP Formulation

$$
\begin{aligned}
\frac{dy_{hid}(u)}{du} &= \frac{d}{du}\left[\frac{1}{1+e^{-u}}\right] \\
&= (1+e^{-u})^{-2}(-e^{-u}) \\
&= \frac{1+e^{-u}-1}{(1+e^{-u})^2} \\
&= \frac{1+e^{-u}}{(1+e^{-u})^2} - \frac{1}{(1+e^{-u})^2} \\
&= \frac{1}{(1+e^{-u})} - \frac{1}{(1+e^{-u})^2} \\
&= \frac{1}{(1+e^{-u})}\left[1-\frac{1}{(1+e^{-u})}\right] \\
&= y_{hid}(u_{hid,1})\left[1-y_{hid}(u_{hid,1})\right]
\end{aligned}
$$



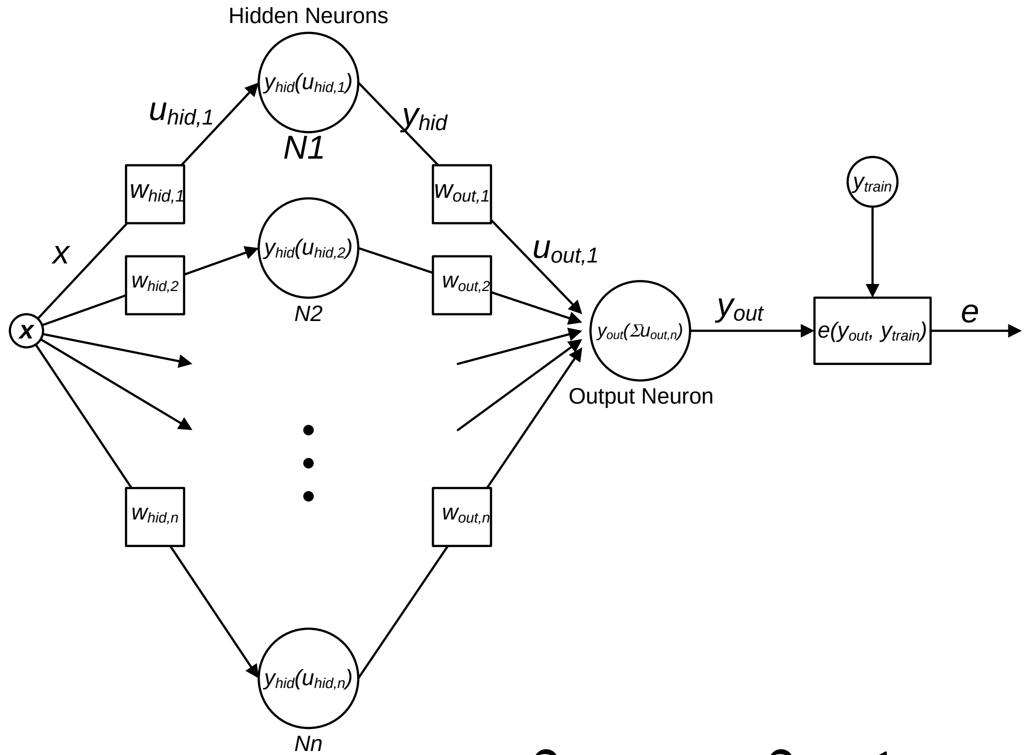
BP Formulation



$$\frac{\partial u_{out,1}}{\partial y_{hid}} = \frac{\partial}{\partial y_{hid}} w_{out,1} y_{hid,1}$$

$$= w_{out,1}$$

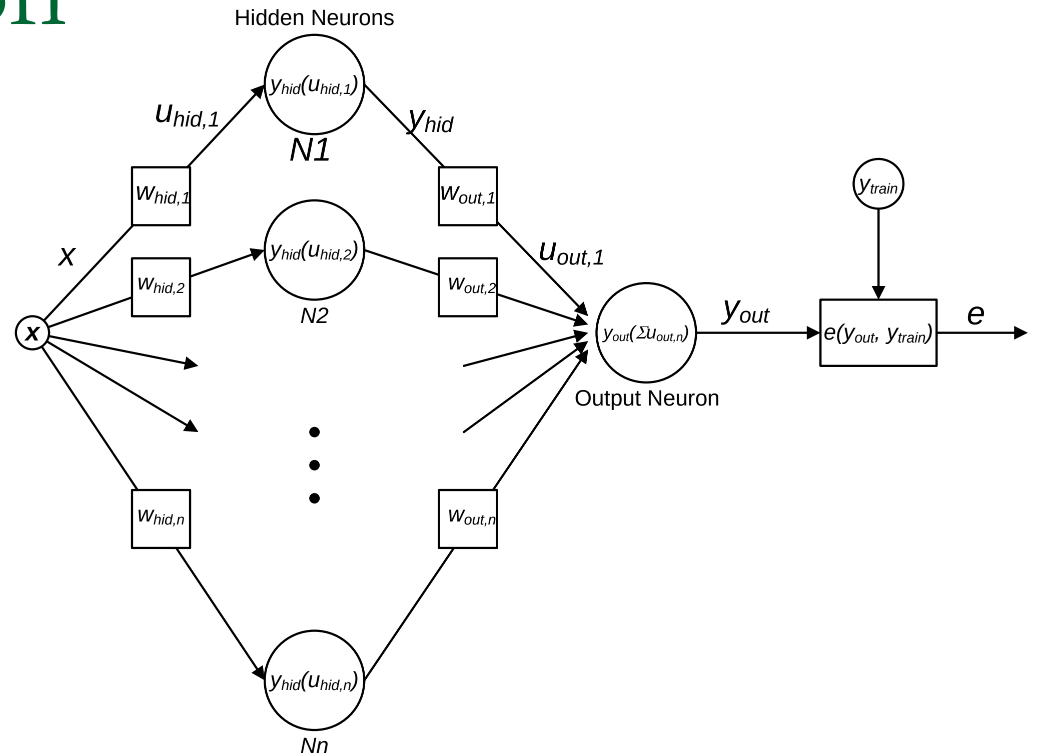
BP Formulation



$$\frac{\partial e}{\partial y_{out}} = \frac{\partial}{\partial y_{out}} \frac{1}{2}(y_{out} - y_{train})^2$$

$$= (y_{out} - y_{train})$$

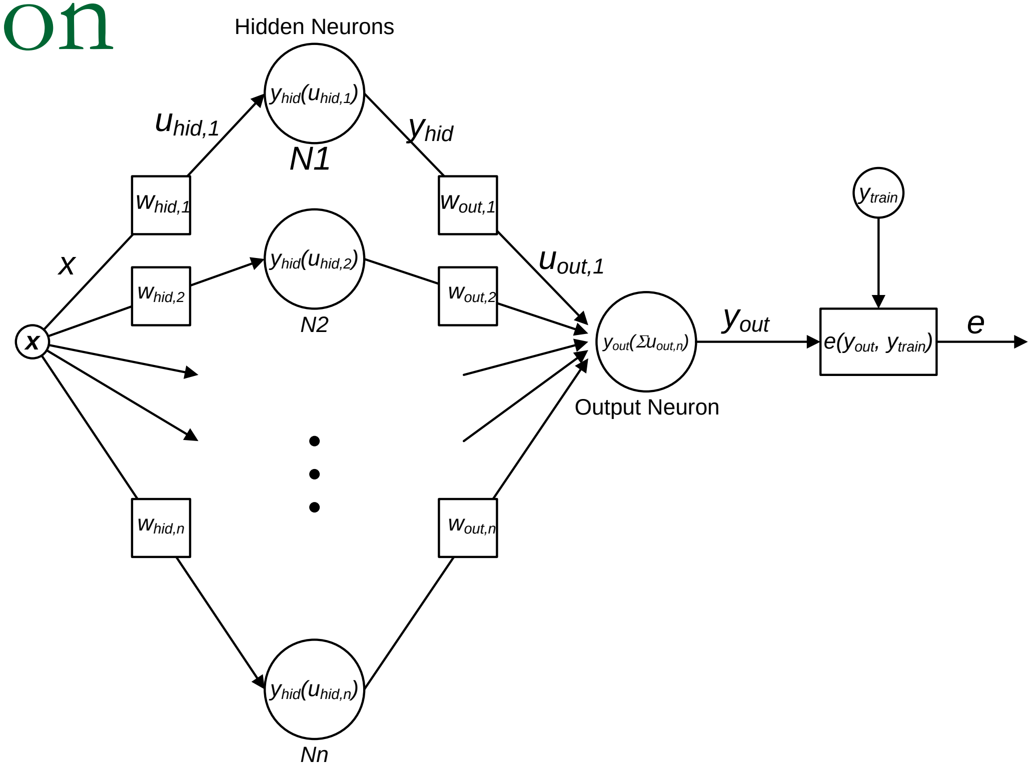
BP Formulation



$$\frac{\partial e}{\partial w_{hid}} = \frac{\partial u_{hid,1}}{\partial w_{hid,1}} \frac{\partial y_{hid}}{\partial u_{hid,1}} \frac{\partial u_{out,1}}{\partial y_{hid}} \frac{\partial y_{out}}{\partial u_{out,1}} \frac{\partial e}{\partial y_{out}}$$

$$= (x)\left(y_{hid}(u_{hid,1})\left[1 - y_{hid}(u_{hid,1})\right]\right)(w_{out,1})(1)(y_{out} - y_{train})$$

BP Formulation



$$\frac{\partial e}{\partial w_{out}} = \frac{\partial u_{out,1}}{\partial w_{out}} \frac{\partial y_{out}}{\partial u_{out,1}} \frac{\partial e}{\partial y_{out}}$$

$$= \left(\frac{\partial}{\partial w_{out}} w_{out,1} y_{hid,1} \right) \left(\frac{\partial}{\partial u_{out,1}} \left[u_{out,1} + u_{out,2} + \ldots + u_{out,N} \right] \right) \left(\frac{\partial}{\partial y_{out}} \frac{1}{2} (y_{out} - y_{train})^2 \right)$$

$$= (y_{hid})(1)(y_{out} - y_{train})$$

Example: The XOR problem:

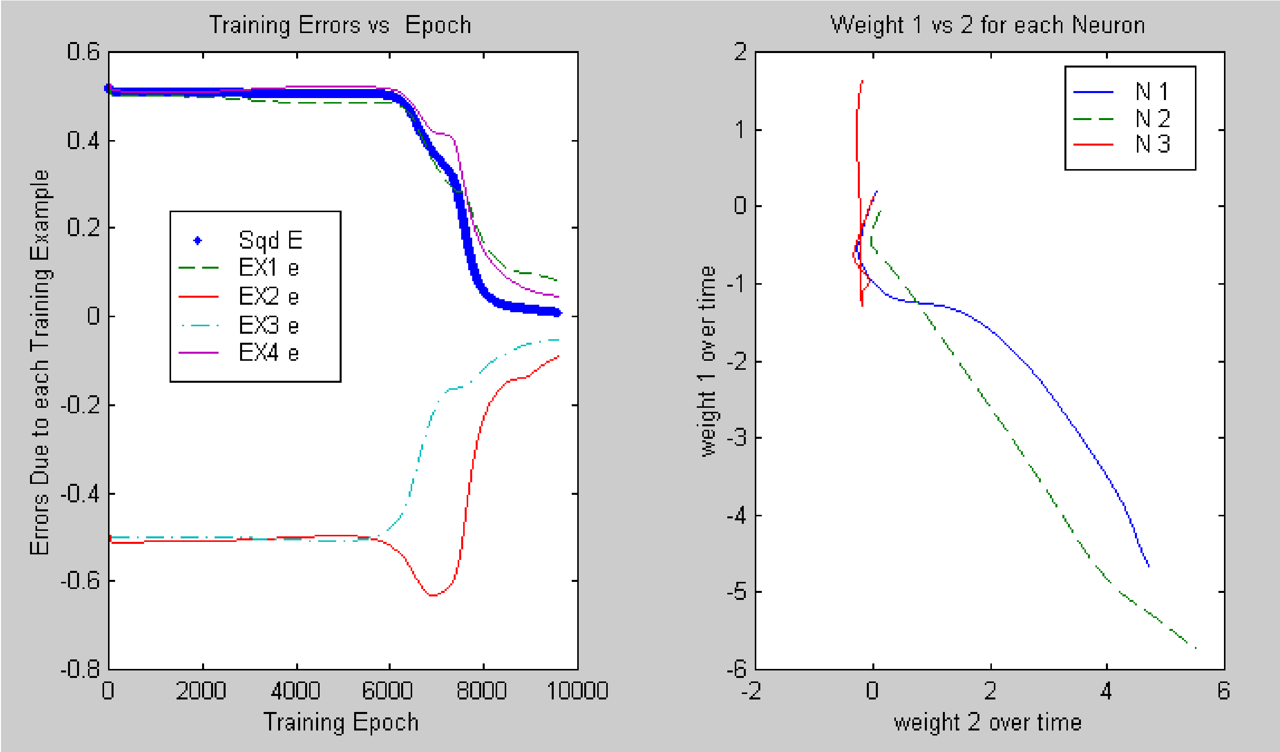
- Single hidden layer: 3 Sigmoid neurons
- 2 inputs, 1 output

	x1	x2	y
Example 1	0	0	**0**
Example 2	0	1	**1**
Example 3	1	0	**1**
Example 4	1	1	**0**

Desired I/O table (XOR):

Example: The XOR problem:

- Training error over epoch



Example: The XOR problem:

initial_weights =
0.0654   0.2017   0.0769   0.1782   0.0243   0.0806
0.0174   0.1270   0.0599   0.1184   0.1335   0.0737
0.1511

final_weights =
4.6970   -4.6585   2.0932   5.5168   -5.7073   -3.2338
-0.1886   1.6164   -0.1929   -6.8066   6.8477   -1.6886
4.1531

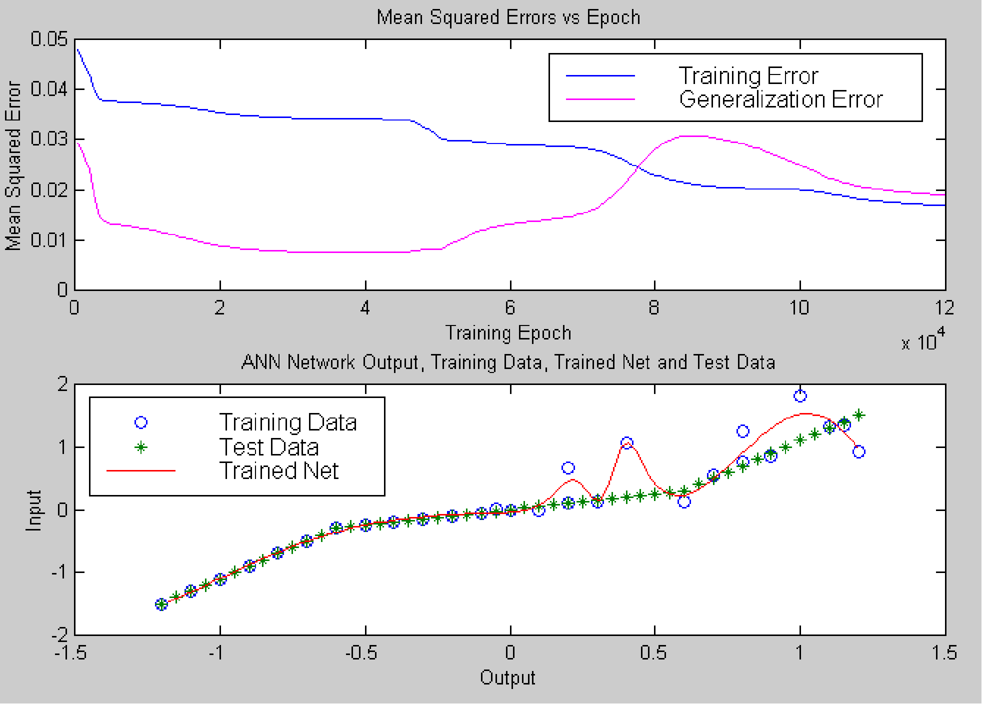
Example: The XOR problem:

Mapping produced by the trained neural net:

	x1	**x2**	**y**
Example 1	0	0	0.0824
Example 2	0	1	0.9095
Example 3	1	0	0.9470
Example 4	1	1	0.0464

Example: Overtraining

- Single hidden layer: 10 Sigmoid neurons
- 1 input, 1 output



Applications of Feed-forward nets

- Pattern recognition
  - Character recognition
  - Face Recognition

- Sonar mine/rock recognition (Gorman & Sejnowksi, 1988)
- Navigation of a car (Pomerleau, 1989)
- Stock-market prediction
- Pronunciation (NETtalk)
  (Sejnowksi & Rosenberg, 1987)